

SISTEM PENGATURAN SUHU RUANGAN YANG ADAPTIF DENGAN *INTEGRASI POSITIONING SYSTEM* BERBASIS WI-FI DAN SENSOR SUHU

Nama Mahasiswa : R Dicky Budi Aldyanto
NRP : 5110100036
Jurusan : Teknik Informatika FTIF-ITS
**Dosen Pembimbing 1 : Waskitho Wibisono, S.Kom.,
M.Eng., Ph.D.**
Dosen Pembimbing 2 : Hudan Studiawan, S.Kom., M.Kom.

Abstrak

Sistem HVAC (Heating, Ventilation, and Air Conditioning) pada bangunan mengeluarkan kurang lebih 50% energi dari total energi yang dikeluarkan oleh bangunan. Pemakaian pendingin ruangan yang tidak terkontrol, seperti halnya membiarkan pendingin ruangan tetap menyala meskipun tidak terdapat seseorang di ruangan tersebut. Dibutuhkan suatu sistem yang dapat mengatur pendingin ruangan untuk mengurangi jumlah energi yang terbuang secara percuma. Sistem tersebut tidak hanya melakukan efisiensi energi, namun juga dapat mengatur suhu ideal di dalam ruangan secara dinamis.

Kondisi pendingin ruangan berubah secara dinamis terkait dengan jumlah orang yang berada di dalam ruangan dan suhu lingkungan. Pendingin ruangan dapat menyala secara otomatis ketika terdapat orang yang berada di dalam, dan akan mati bila ruangan kosong. Suhu pada pendingin ruangan berubah secara dinamis mengikuti naik dan turunnya suhu lingkungan, namun juga ditetapkan batas atas dan batas bawah suhu pendingin ruangan untuk mencegah suhu di dalam ruangan terlalu dingin

maupun terlalu panas. Mikrokontroler Arduino tersambung pada sensor suhu dan pengendali pendingin ruangan jarak jauh yang berfungsi memberikan perintah pada pendingin ruangan.

Data jumlah orang yang terdapat di dalam ruangan diperoleh dari Indoor Positioning System dan data suhu lingkungan digunakan server untuk menentukan suhu ideal secara dinamis di dalam ruangan dan perintah yang akan dikirimkan pada mikrokontroler melalui jaringan WLAN. Pengguna dapat melihat data informasi terkini melalui aplikasi web dan mobile.

Sistem yang telah dibangun kemudian diuji coba mulai dari uji fungsionalitas hingga uji performa dengan menggunakan beberapa skenario yang telah ditentukan. Berdasarkan uji coba tersebut, dapat disimpulkan bahwa sistem telah mampu memerintah pendingin ruangan dan memberikan suhu secara dinamis dengan integrasi jumlah orang di dalam ruangan dan suhu lingkungan.

Kata kunci: Smart Room, Suhu Adaptif, Mikrokontroler, Arduino, Indoor Positioning System.

ADAPTIVE ROOM TEMPERATURE REGULATION SYSTEM WITH INTEGRATUIN OF INDOOR POSITIONING SYSTEM BASED ON WI-FI AND TEMPERATURE SENSORS

Student's Name : R Dicky Budi Aldyanto
Student's ID : 5110100036
Department : Informatics Engineering, FTIF-ITS
First Advisor : Waskitho Wibisono, S.Kom.,
M.Eng., Ph.D.
Second Advisor : Hudan Studiawan, S.Kom., M.Kom.

Abstract

HVAC (Heating, Ventilation, and Air-conditioning) systems in buildings use approximately 50% energy from the total energy consumed by the building. For example, uncontrolled use of air conditioning as well as allowing the AC stays on when no one in the room. From the above problems, we need a system that can automatically adjust the air conditioning in order to reduce the amount of energy wasted in vain. The system not only can improve efficiency of energy, but also can adjust the ideal indoors temperature dynamically.

The state of air conditioner change dynamically depends on the amount of a person that was in the room, as well as environmental temperature. Air conditioner can be turned on automatically when there is a person inside, and will automatically turns off when the room is empty. Air conditioner temperature can change dynamically depends on the surrounding environment. Air conditioner temperatures will be set upper limit and lower limit to prevent the temperature in the room becomes too hot or too cold. Arduino microcontroller is connected to a temperature sensor and an AC remote controller that serves as a conduit command on the AC.

Information about the number of a person that were in the room obtained from the mechanism of the indoor positioning system. Environmental data temperature used by the server to dynamically determine the ideal temperature in the room, the data is also used to specify a command to be sent to the microcontroller via a WLAN network. Users can see recent data information through the web or mobile application.

The system that has been built will be tested ranging from the functionality test to performance test by using pre-determined scenarios. Based on the tests that have been done, concluded that the system can adjust the air conditioning and provide temperature dynamically with the integration of total amount of person inside the room and environmental temperature.

Keywords: Smart Room, Adaptive Temperature, Microcontroller, Arduino, Indoor Positioning System.

BAB II

TINJAUAN PUSTAKA

Bab ini berisikan mengenai teori-teori yang berkaitan dengan pengimplementasikan perangkat lunak, dengan tujuan memberikan gambaran secara umum mengenai sistem yang dibangun dan berguna dalam pengembangan Tugas Akhir ini.

2.1. Mikrokontroler Arduino

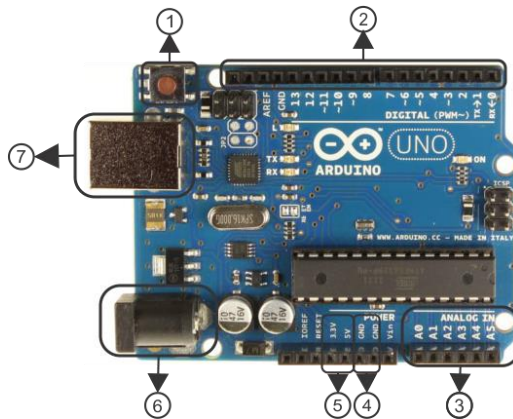
Arduino [2] merupakan mikrokontroler papan tunggal berbasis processor *Atmel AVR* yang bersifat *open source*. Mikrokontroler diprogram menggunakan Arduino IDE (*Integrated Development Enviroment*) yang terdiri dari berbagai macam *library*. Bahasa pemograman yang digunakan adalah bahasa C. Tersedia berbagai macam jenis mikrokontroler Arduino, seperti: Arduino Uno, Arduino Leonardo, dan lainnya. Arduino Uno merupakan jenis yang sering digunakan pada umumnya. Mikrokontroler ini terdapat *input* serta *output* yang berguna dalam menerima masukan dari berbagai macam sensor dan mengendalikan lampu, motor, atau *actuator*. Salah satu sensor yang dapat digunakan adalah sensor LM35, dimana sensor tersebut dapat mendeteksi besaran nilai suhu ruangan.

Arduino juga dapat berkomunikasi dengan komputer maupun sesama mikrokontroler menggunakan kabel USB ataupun dengan *shield*. Wi-Fi *shield* merupakan salah satu *actuator* yang berfungsi untuk melakukan koneksi antar perangkat, computer, maupun internet menggunakan *wireless*.

2.2. Arduino Uno

Arduino Uno [3] merupakan sebuah papan mikrokontroler yang berbasis pada ATmega328, dapat dilihat pada Gambar 2.1. Memiliki 14 *pin* untuk *input* atau *output*, enam *input* analog, 16

MHz resonator keramik, koneksi terhadap USB, *power jack*, *ICSP header*, dan tombol untuk *reset*, untuk mendukung kinerja mikrokontroler. Arduino Uno mudah dalam penggunaannya, pengguna hanya diperlukan menghubungkan ke komputer menggunakan Kabel USB, adaptor AC-DC atau baterai. Mikrokontroler ini berbeda dengan papan mikrokontroler yang lain dikarenakan tidak menggunakan *FTPI chip driver USB-to-serial*.



Gambar 2.1. Arduino Uno [3].

1. *Tombol Reset*
Merupakan sebuah tombol yang memiliki fungsi untuk melakukan *reset* terhadap program yang sudah diunggah pada mikrokontroler.
2. *Digital Input/Output Pins*
Pin tersebut berfungsi sebagai penghubung antar komponen digital, dimana enam diantara 14 *pin* dapat digunakan sebagai *PWM outputs*.
3. *Analog Input Pins*
Enam *input pin* analog yang berfungsi dalam penerima data analog dari berbagai komponen. Sebagai contoh, digunakan untuk menerima data dari sensor suhu LM35.
4. *Ground Pin*

Pin yang berfungsi menghubungkan *pin ground* Arduino dengan komponen seperti rangkaian sensor.

5. *VCC Pin*

Pin yang menghantarkan tegangan sebesar tiga koma tiga dan lima voltase untuk mensuplai tegangan untuk komponen seperti rangkaian sensor.

6. *Power Jack*

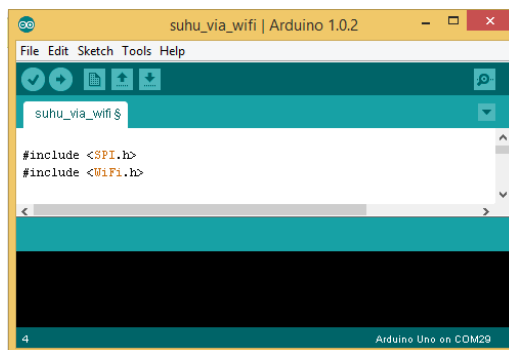
Merupakan soket yang berfungsi mensuplai tegangan sebesar sembilan *volt* pada mikrokontroler.

7. *USB Connection*

Merupakan soket kabel USB untuk menggunggah kode program dari komputer kepada Arduino.

2.3. Arduino Wi-Fi Shield

Shield [4] merupakan papan ekstensi yang dapat dipasang untuk meningkatkan kemampuan sebuah mikrokontroler Arduino. Ethernet shield merupakan salah satu contoh *shield* yang tersedia untuk Arduino, dimana berfungsi melakukan koneksi menggunakan LAN (*Local Area Network*).

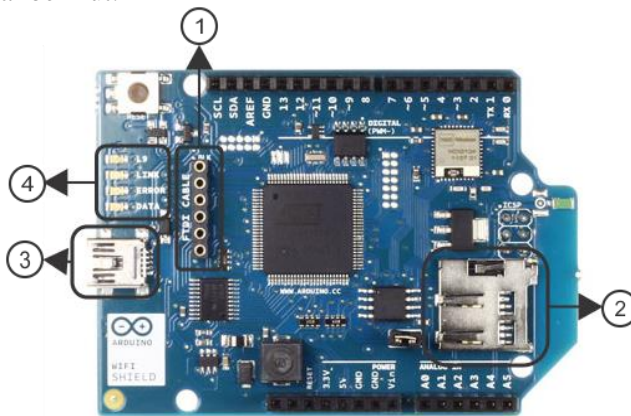


Gambar 2.2. Pustaka Wi-Fi.

Arduino juga dapat terhubung dengan internet tanpa terhubung dengan kabel, menggunakan teknologi IEEE 802.11

atau Wi-Fi berbasis sistem HDG104 Wireless LAN 802.11b/g. Arduino Wi-Fi Shield terdiri dari mikrokontroler Atmega 32UC3 yang memungkinkan pemakaian protokol TCP dan UDP dalam mengirimkan paket data.

Pustaka Wi-Fi diperlukan pada penulisan kode program untuk dapat mengirimkan paket data menggunakan WLAN (*Wireless Local Area Network*) seperti pada Gambar 2.2. Pada Gambar 2.3 menunjukan komponen-komponen yang terdapat pada Arduino Wi-Fi Shield. Komponen-komponen tersebut dijelaskan sebagai berikut.



Gambar 2.3. Arduino WiFi Shield [4].

1. *FTDI Connector*
FTDI connector memungkinkan komunikasi serial untuk mendiagnosa komunikasi.
2. Micro-SD slot
Tempat meletakkan kartu Micro-SD yang digunakan untuk penyimpanan berkas yang melayani komunikasi antar jaringan dengan menggunakan pustaka kartu SD.
3. Micro USB
Merupakan slot kabel USB yang berfungsi untuk memperbarui kode instruksi yang tertanam secara permanen pada *read-only memory*.

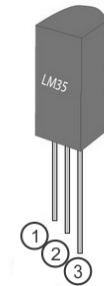
4. Indikator LED

Indikator LED berfungsi memberikan informasi mengenai kondisi *shield*.

2.4. Sensor Suhu (LM35)

Sensor suhu LM35 [5] pada Gambar 2.4 merupakan sirkuit sensor suhu terintegrasi secara presisi, dimana hasil keluaran voltase berbanding lurus dengan temperatur Celsius. Kelebihan LM35 adalah sensor terkalibrasi pada derajat suhu Kelvin, sehingga pengguna tidak memerlukan pengurangan besar tegangan dalam mendapatkan skala Celsius. LM35 juga tidak memerlukan tambahan kalibrasi untuk memperoleh akurasi hasil, dimana tingkat akurasi pada suhu ruangan ialah $\pm 0.4^{\circ}\text{C}$ dan $\pm 0.8^{\circ}\text{C}$. Rentang suhu yang terdeteksi sensor antara 0°C hingga 100°C .

Sensor ini memberikan hasil lebih akurat dibandingkan dengan thermistor, serta menghasilkan tegangan *output* yang lebih tinggi dari *thermocouples*.



Gambar 2.4. Sensor Suhu LM35 [6] .

1. VCC Pin

Pin pada sensor LM35 terhubung dengan *pin* vcc pada mikrokontroler Arduino untuk memperoleh sumber tegangan.

2. Output Pin

Pin ini terhubung dengan *pin input* analog pada mikrokontroler Arduino untuk memberikan hasil pendeteksian suhu udara secara analog.

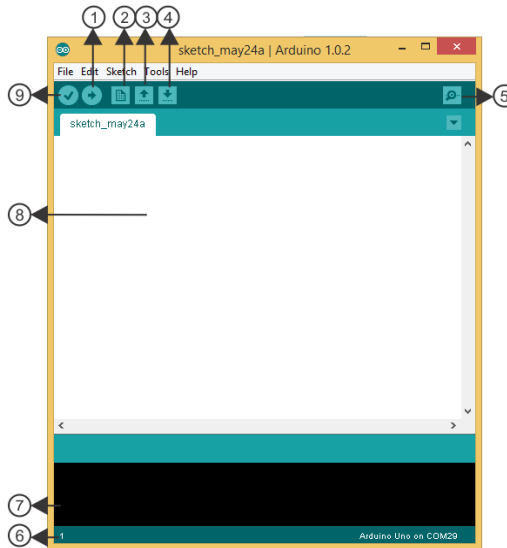
3. *Ground Pin*

Ground pin terhubung dengan *pin ground* pada mikrokontroler Arduino.

2.5. Arduino IDE

Arduino IDE [2] (*Integrated Development Enviroment*) merupakan program komputer untuk menulis kode pemrograman dengan menggunakan bahasa pemrograman C/C++, dimana kode tersebut akan diunggah pada mikrokontroler Arduino dalam bentuk berkas HEX untuk memberikan instruksi ataupun perintah yang diinginkan oleh pengguna. Arduino IDE dapat dilihat pada Gambar 2.5.

Mikrokontroler Arduino dihubungkan dengan PC menggunakan kabel USB, namun pengguna harus melakukan instalasi FTDI USB *driver* supaya PC dapat mendeteksi Arduino melalui serial *port* tertentu dimana Arduino tersebut tersambung. Setelah Arduino tersambung dengan komputer, terlebih dahulu harus menentukan *board* Arduino dan serial *port* yang akan digunakan selama proses pengkodean, seperti pada Gambar 2.6 dan Gambar 2.7. Terdapat berbagai macam jenis *board* yang ditampilkan oleh Arduino IDE. Contohnya, untuk menggunakan papan mikrokontroler Arduino Uno, maka pada saat memilih *board* pada Arduino IDE, memilih pilihan Arduino Uno untuk menghindari terjadinya kesalahan. Apabila terjadi kesalahan dalam memilih *port* untuk Arduino maka Arduino IDE akan menampilkan pesan kesalahan yang sedang terjadi.



Gambar 2.5. Arduino IDE.

1. Tombol Upload
Tombol ini berfungsi untuk menggunggah *sketch* atau kode program pada mikrokontroler Arduino.
2. Tombol New Sketch
Tombol New Sketch merupakan tombol untuk membuat berkas *sketch* yang baru.
3. Tombol Open
Tombol Open berguna untuk membuka berkas *sketch* yang telah dibuat sebelumnya.
4. Tombol Save
Tombol Save memiliki fungsi untuk menyimpan *sketch* atau kode program yang telah disusun.
5. Tombol Serial
Tombol ini berfungsi untuk menampilkan komunikasi serial antara mikrokontroler Arduino dengan PC.
6. Line Number

Memberikan informasi mengenai nomor kolom kode program yang sedang disusun.

7. Konsol Teks

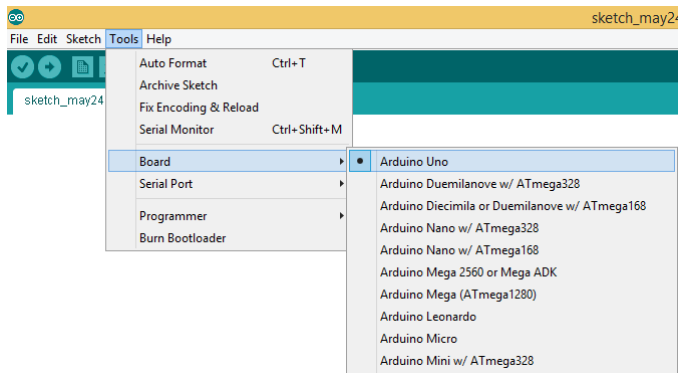
Konsol ini berfungsi memberikan informasi mengenai status ataupun pesan kesalahan yang terjadi ketika proses pengkodean dilakukan.

8. Sketch Editor

Merupakan tempat untuk menyusun kode pemrograman.

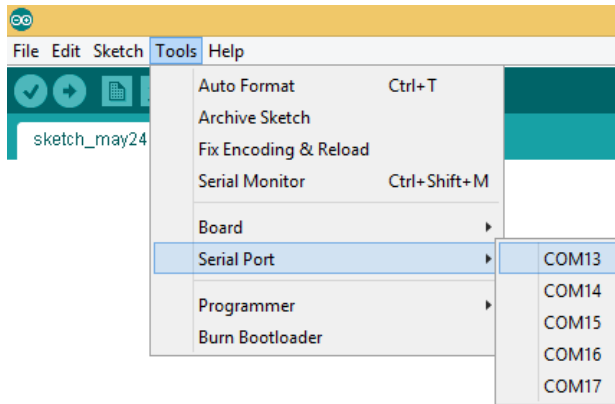
9. Tombol Verify

Tombol Verify merupakan tombol untuk melakukan verifikasi kode pemrograman untuk menemukan kesalahan pada kode program.



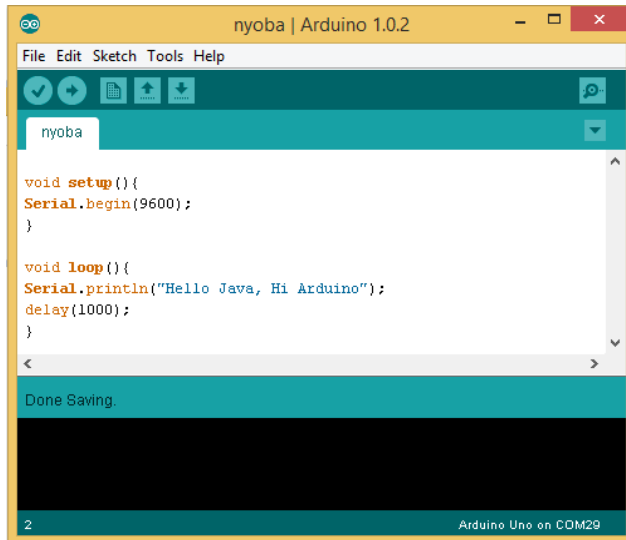
Gambar 2.6. Memilih jenis *board* Arduino.

Pada dasarnya terdapat dua fungsi utama dalam proses pemrograman mikrontroler Arduino yaitu fungsi *setup* dan *loop* seperti pada Gambar 2.8, dimana fungsi *setup* berguna untuk melakukan insialisasi pada saat pertama kali kode program dijalankan pada mikrokontroler Arduino. Sedangkan fungsi *loop* berfungsi melakukan proses pengulangan secara berulang-ulang kali ketika mikrokontroler Arduino dinyalakan.



Gambar 2.7. Memilih port untuk Arduino.

Namun sebelum program dapat berjalan pada mikrokontroler, kita perlu melakukan pengunggahan kode pemrograman kepada mikrokontroler dengan menekan tombol Upload. Untuk dapat mengetahui hasil *output* dari mikrokontroler Arduino, kita dapat membuka Serial Monintor dengan menekan tombol Serial yang terdapat pada bagian sebelah kanan Arduino IDE. Terdapat keterbatasan memori pada mikrokontroler Arduino yang memiliki memori sebesar 32,256 *byte* atau 32 Kb, maka berkas *sketch* atau kode program yang akan diunggah tidak boleh melebihi daya tampung dari memori. Apabila kode program yang kita susun melebihi batas maksimal, maka kita tidak dapat menanamkan berkas *sketch* atau kode program pada mikrokontroler.



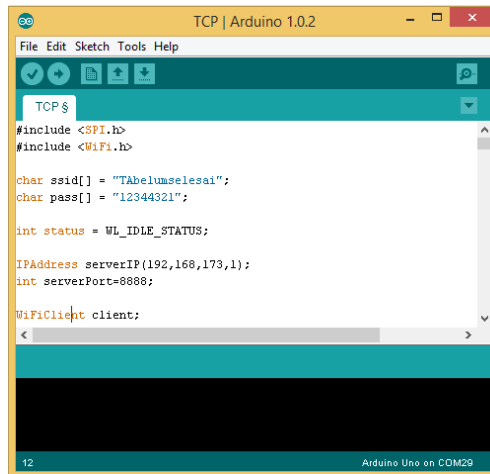
Gambar 2.8. Fungsi *Setup* dan *Loop*.

2.6. Arduino dan Java menggunakan protokol TCP

Arduino [7] merupakan sebuah mikrokontroler yang menggunakan bahasa pemrograman C dalam proses penyusunan kode. Dengan adanya ekstensi berupa Wi-Fi Shield, mikrokontroler Arduino dapat menjadi sebuah klien pada proses komunikasi jaringan klien-*server*. Dimana *server* dapat menggunakan bahasa pemrograman Java, sedangkan mikrokontroler menggunakan bahasa pemrograman C. Hal ini tentu meningkatkan fleksibilitas proses pengkodean pada sebuah jaringan klien-*server* untuk dapat melakukan pengiriman paket data dari mikrokontroler Arduino terhadap *server* ataupun sebaliknya.

Proses pengiriman paket data dalam jaringan klien-*server* pada mikrokontroler Arduino dan Java *server* menggunakan TCP (*Transmission Control Protocol*). TCP merupakan protokol komunikasi yang umum digunakan, dimana paket data yang

diterima dalam bentuk yang sempurna dengan menggunakan *layer IP Network*. Dengan menggunakan *header*, dimana sumber dan tujuan alamat jaringan yang tertuju menjadi satu dengan paket data yang dikirim menggunakan TCP dapat dikirim ke seluruh dunia menggunakan internet. Hal yang utama dilakukan dalam proses pengiriman data menggunakan Wi-Fi (*Wireless Fidelity*) dengan protokol TCP adalah inisialisasi SSID (*Service Set Identification*) serta *password* untuk dapat terhubung secara *wireless*, IP (*Internet Protocol*) tujuan dan *port* yang akan digunakan pada mikrokontroler Arduino sebagai klien, seperti pada Gambar 2.9.



Gambar 2.9. Inisialisasi pada mikrokontroler Arduino.

Pada *java server* perlu dilakukan inisialisasi *port* yang akan digunakan, sesuai dengan yang diinisialisasikan pada klien untuk dapat berkomunikasi, seperti pada Gambar 2.10. Selanjutnya *server java* dapat menerima data dari klien maupun mengirim paket data kepada klien.

```

package com.main;

import java.io.*;

public class main {

    private static final int port=8888;

    public static void main(String argv[]) throws Exception
    {
        String clientMsg;
        ServerSocket mySocket = new ServerSocket(port);
        System.out.println("TCP-Server started");

        while(true)
        {
            Socket connectionSocket = mySocket.accept();
            BufferedReader clientInput = new BufferedReader(new InputStreamReader(conn
            DataOutputStream clientOutput = new DataOutputStream(connectionSocket.getO
            clientMsg = clientInput.readLine();
            System.out.println("Client says: " + clientMsg);
            clientMsg = "You said:"+clientMsg + '\n';
            clientOutput.writeBytes(clientMsg);
        }
    }
}

```

Gambar 2.10. Inisialisasi pada Java server.

2.7. Android SDK

Android SDK [8] (*Software Development Kit*) adalah perangkat lunak yang digunakan oleh para pengembang aplikasi dalam membangun atau *development* aplikasi untuk *platform* Android. Sampel proyek termasuk di dalam SDK tersebut disertai dengan kode program. *Software Development Kit* terdiri dari perangkat lunak untuk pengembangan, emulator, dan pustaka yang diperlukan dalam mengembangkan aplikasi Android menggunakan bahasa pemrograman Java. Emulator pada Android SDK memungkinkan developer melakukan uji coba perangkat lunak yang dibuat tanpa telepon seluler berbasis Android.

Google juga menyediakan ADT (*Android Developer Tools*) untuk membantu pengembang dalam mengembangkan aplikasi Android, dimana ADT merupakan satu set komponen ekstensi bagi Eclipse IDE. ADT terdiri dari semua fungsionalitas yang dibutuhkan dalam membuat, menyusun, dan mengembangkan aplikasi Android dari Eclipse IDE. ADT menyediakan sejumlah fitur, dimana pengembang dapat melakukan desain serta membangun tampilan aplikasi yang menarik dengan cara mengakses berkas XML *layout* pada Eclipse.

2.8. Algoritma *Cluster Filtered K-Nearest Neighbors* (CFK)

Langkah awal algoritma *Cluster Filtered K-Nearest Neighbors* [9] adalah mendapatkan *K neighbors*, hal ini sama dengan *KNN*. Namun, tidak seperti *KNN* yang mengambil semua *K neighbors* terdekat untuk diperhitungkan hasil estimasi. *CFK* menggunakan teknik pengelompokan dalam pembagian tetangga menjadi beberapa kelompok dan hanya satu kelompok yang dipilih sebagai delegasi, sementara yang lain dikeluarkan. Langkah-langkah tersebut dapat dilihat pada Gambar 2.11.

- | | |
|---|---|
| 1 | Go through the sample space S , find the set K_Set_L into several <i>clusters</i> |
| 2 | Apply some clustering algorithm CF to K_Set_L , partition K_Set_L into several <i>clusters</i> |
| 3 | According to the selecting rule set R , select one <i>Cluster</i> as delegate, say C ; |
| 4 | Take the average of the coordinates (x,y) of all samples in C as the estimate the location (x,y) : |
| | $L.x = \frac{\sum_{SPi \in C} SPi.x}{ C } \quad L.y = \frac{\sum_{SPi \in C} SPi.y}{ C }$ |
| | ps : C stands for the cardinality of C |

Gambar 2.11. *Clustering Filtered KNN* [9].

2.9. *Fingerprint*

Fingerprint [10] digunakan untuk mendapatkan perkiraan mengenai titik letak posisi setelah dilakukan kalibrasi. Terdapat dua tahap menentukan lokasi *fingerprint*, yaitu tahap kalibrasi atau dikenal dengan tahap *offline* dan tahap menentukan lokasi atau tahap *online*. Tahap kalibrasi menggunakan perangkat bergerak untuk menghitung nilai RSS. Setiap *n* dari perhitungan menjadi bagian yang disebut dengan *tuple* (*q_i, r_i*) *i* = 1,2,...,n dimana *q_i* =

(x_i, y_i) yang merupakan koordinat geografi dari lokasi ke I dan $r_i = (r_{i1}, r_{i2}, \dots, r_{im})$ merupakan nilai dari RSS. Umumnya, nilai rata-rata dari sejumlah data contoh disimpan secara masing-masing lokasi.

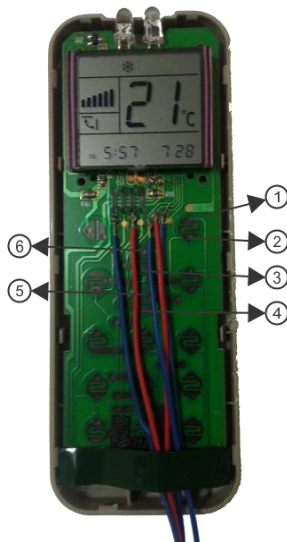
2.10. Pengendali Jarak Jauh Universal pada Pendingin Ruangan

Pengendali jarak jauh universal [11] untuk pendingin ruangan atau yang sering disebut dengan *remote air conditioner* pada Gambar 2.12 merupakan perangkat nirkabel untuk melakukan pengaturan terhadap pendingin ruangan. Perangkat ini berguna untuk menghidupkan dan mematikan pendingin ruangan, menaikkan serta menurunkan suhu ruangan. Dalam pengiriman data pada pengendali jarak jauh melalui IR (*infrared*) dibutuhkan pemancar dan penerima. Pada dasarnya penggunaan IR pada perangkat nirkabel ini didasari oleh konsumsi tenaga yang rendah, pengoperasian dengan voltase rendah, hemat biaya, dan mudah dimodifikasi. IR merupakan radiasi elektromagnetik dengan panjang gelombang 0.7 dan 300 mikrometer dan memiliki bentang frekuensi sebesar satu dan 430 THz.



Gambar 2.12. Pengendali jarak jauh pendingin ruangan.

Pada saat kita menekan tombol untuk menghidupkan, mematikan pendingin ruangan, serta menaikkan dan menurunkan suhu ruangan, hal yang terjadi pada pengendali jarak jauh adalah mempertemukan dua sisi penghantar tegangan untuk dapat memberikan perintah pada pendingin ruangan. Tampak bagian dalam dari pengendali jarak jauh pendingin ruangan dapat dilihat pada Gambar 2.13.



Gambar 2.13. Bagian dalam pengendali jarak jauh pendingin ruangan.

1. Sisi nomor satu
Sisi ini berfungsi untuk menghidupkan maupun mematikan pendingin ruangan apabila sisi ini dihubungkan dengan sisi nomor dua.
2. Sisi nomor tiga
Sisi pada nomor tiga memiliki fungsi dalam memberikan perintah kepada pendingin ruangan untuk menaikkan suhu ruangan apabila sisi ini dihubungkan dengan sisi nomor enam.
3. Sisi nomor empat

Sisi pada nomor empat ini berfungsi dalam memberikan perintah kepada pendingin ruangan untuk menurunkan suhu ruangan.

2.11. Indoor Positioning System (IPS)

IPS [1] berbasis pada konsep *Context-Aware*, dimana sekumpulan informasi dapat dipergunakan untuk menggambarkan situasi titik letak pada suatu tempat. Sistem ini menggunakan teknologi nirkabel atau *wireless* yang pada umumnya diterapkan untuk menemukan seseorang maupun barang di dalam sebuah ruangan ataupun gedung. Informasi titik letak yang diperoleh pada sistem ini lebih efektif dibanding penggunaan *Global Positioning System* untuk kondisi dimana seseorang ataupun barang berada di dalam ruangan atau gedung, dikarenakan keterbatasan gelombang sinyal.

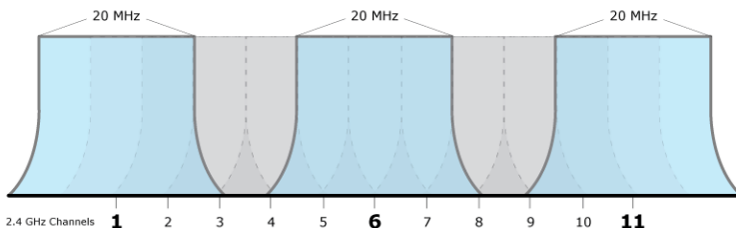
Penggunaan *Wireless Ethernet IEEE 802.11* (Wi-Fi) pada IPS memiliki kelebihan dibandingkan teknologi lainnya, yaitu biaya yang cenderung murah. Sinyal radio yang dipancarkan pada Wi-Fi dapat digunakan untuk melakukan estimasi letak posisi menggunakan nilai terukur yang didapat pada perangkat bergerak dengan fasilitas Wi-Fi. Oleh karena itu, IPS digunakan pada Tugas Akhir ini untuk melakukan estimasi jumlah orang yang berada di dalam ruangan.

2.12. Wi-Fi

Wireless Fidelity [12] (Wi-Fi) mengacu pada *IEEE 802.11 Wireless Local Area Network* merupakan teknologi berbasis jaringan nirkabel atau tanpa kabel yang digunakan untuk menghubungkan komputer atau perangkat lainnya, seperti laptop, ponsel *pintar*, dan lain sebagainya. Teknologi ini menjadi populer seiring perkembangan konektivitas IP pada jaringan kantor dan kampus. Terdapat lima macam variasi dari Wi-Fi berdasarkan pada asosiasi IEEE, yaitu:

1. 802.11, merupakan versi pertama yang mendukung transmisi sebesar 1 hingga 2 Mbps menggunakan frekuensi 2.4 GHz.
2. 802.11b, merupakan perkembangan dari 802.11 dengan transmisi hingga sebesar 11 Mbps menggunakan frekuensi 2.4 GHz.
3. 802.11a, menggunakan frekuensi 5 GHz dengan transmisi sebesar 54 Mbps.
4. 802.11g, merupakan perkembangan dari 802.11 yang meningkatkan transmisi hingga 54 Mbps dengan frekuensi 2.4 GHz.
5. 802.11n, merupakan hasil modifikasi pada *layer physical* dan *MAC* untuk meningkatkan dukungan kemampuan hingga 100 Mb/s dengan frekuensi mencapai 5 GHz.

Wi-Fi [13] memiliki spektrum sebesar 100 MHz yang terbagi menjadi 11 saluran, setiap saluran memiliki lebar spektrum sebesar 20 hingga 22 MHz. Interferensi sering kali terjadi pada Wi-Fi yang letaknya berdekatan dan memiliki saluran dengan frekuensi yang bersamaan, seperti saluran 9 akan terjadi interferensi pada saluran 7, 8, 10, dan 11. Oleh karena itu pemilihan saluran merupakan hal yang penting, supaya tidak terjadi *overlapping* atau interferensi. Saluran 1, 6, dan 11 pada Gambar 2.14 tidak akan terjadi interferensi dikarenakan terdapat jeda antar saluran sebesar 20 MHz.



Gambar 2.14. Saluran *non-overlapping* pada Wi-Fi .

2.13. PHP

*Personal Home Page*¹ (PHP) merupakan bahasa pemrograman *open source* yang berjalan pada *webserver*. PHP dibangun untuk membuat konten yang dinamis pada sebuah website digunakan membuat dokumen XML, melakukan pengolahan data, seperti melakukan proses menampilkan, memperbarui, memasukan, dan menghapus data yang terdapat pada *database*. Bahasa ini digunakan untuk membangun sebuah *webservice* dalam membuat dokumen XML dan menampilkan informasi.

Sistem kerja PHP diawali dengan melakukan permintaan (*request*) oleh klien dengan mengakses alamat URL pada *web browser*, dimana *server* akan melakukan proses identifikasi pada halaman yang dicari dan memeriksa apakah file tersebut mengandung *script* PHP. Bahasa pemrograman ini digunakan sebagai *end-user* untuk menampilkan data gambar berupa letak posisi pengguna, dan data sistem pendingin terkini.

2.14. WebService

WebService [14] merupakan kumpulan protokol dan metode untuk pertukaran data antara aplikasi dengan sistem. Aplikasi perangkat lunak tertulis menggunakan bahasa pemrograman yang beranekaragam dan berjalan di berbagai lintas *platform* yang dapat menggunakan *webservice* dalam pertukaran data antar komputer pada internet. Data yang dikirim oleh *server* memiliki format XML yang kemudian diolah oleh ponsel *pintar*. Protokol HTTP digunakan untuk mengirimkan dokumen XML kepada aplikasi *end-user*. *Webservice* sistem ini juga berguna untuk menyimpan data pada *database*.

¹ PHP. www.php.net/

BAB III

PERANCANGAN PERANGKAT LUNAK

Bab ini akan menjelaskan mengenai dasar dari perancangan perangkat lunak pada sistem yang akan dibangun pada Tugas Akhir ini. Perancangan merupakan hal yang penting dalam proses pembuatan perangkat lunak dimana akan dijelaskan secara khusus pada bab ini mengenai deskripsi umum aplikasi, proses perancangan, alur dan implementasinya.

3.1. Deskripsi Umum Sistem

Pada Tugas Akhir ini akan dibangun sistem suhu adaptif dengan integrasi *indoor positioning system* berbasis Wi-Fi dan sensor suhu. Perangkat lunak dengan studi kasus ruangan laboratorium pada sistem ini diharapkan dapat menghidupkan, mematikan pendingin ruangan, menaikkan, serta menurunkan suhu ruangan dengan *trigger* yang tersedia. Data yang diperlukan pada sistem ini adalah estimasi jumlah orang yang berada di ruangan, dan besaran nilai suhu yang terdapat di luar ruangan.

Perangkat lunak pada sistem ini berbasis klien-*server*, dimana *server* akan melakukan estimasi jumlah orang dan menentukan suhu ideal pada suatu ruangan. Suhu pada ruangan bersifat adaptif terkait dengan jumlah orang dan besaran nilai suhu yang didapat oleh sensor suhu yang terhubung pada mikrokontroler Arduino.

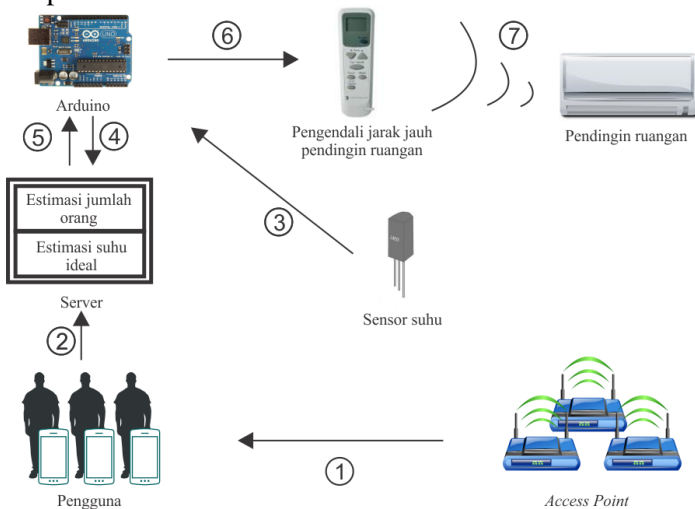
Disisi pengguna terdapat perangkat lunak berbasis Android yang berfungsi melakukan *scanning* terhadap sinyal *access point* dan mengirimnya pada *server* melalui *webservice* untuk dapat dilakukan pemetaan letak lokasi untuk mengetahui apakah pengguna tersebut berada di dalam ruangan atau tidak. Setelah ditemukan data titik letak posisi pengguna, maka dilakukan estimasi jumlah orang. Selanjutnya *server* menentukan suhu ideal pada ruangan dengan data yang ada dan menyimpan data terkini di *database*. Apabila terdapat seseorang atau lebih pada ruangan

tersebut, maka *server* melakukan respon terkait *trigger* yang tersedia dengan mengirimkan perintah terhadap mikrokontroler Arduino melalui WLAN. Kemudian mikrokontroler tersebut memberikan perintah terhadap pengendali jarak jauh untuk menghidupkan pendingin ruangan.

Pada sisi *end-user* dari sistem ini terdapat sebuah beranda berbasis aplikasi web dan perangkat lunak pada telepon seluler pengguna yang dimana menampilkan data terkini terkait titik letak lokasi pengguna, dan kondisi terkini daripada sistem tersebut.

3.2. Arsitektur Umum Sistem

Rancangan arsitektur pada sistem yang akan dibangun dapat dilihat pada Gambar 3.1.



Gambar 3.1. Arsitektur umum sistem.

Berdasarkan pada Gambar 3.1, estimasi jumlah orang ditentukan oleh data kekuatan sinyal yang diterima. *Access point* memancarkan gelombang radio ke sekitarnya. Pada sisi pengguna,

kekuatan sinyal dari gelombang tersebut *dipindai* selama beberapa saat untuk dikumpulkan menjadi sebuah data sinyal. Data sinyal kemudian disimpan untuk dikirimkan kepada *server*.

Server dan telepon selular pengguna terhubung pada jaringan yang sama, pada sistem ini akan digunakan jaringan WLAN dimana *server* dan telepon selular akan terhubung pada *access point* yang sama. Selanjutnya, data tersebut dikirim kepada *server* untuk dilakukan pemetaan titik lokasi pengguna. *Server* melakukan proses *listening* tanpa henti untuk menunggu data yang akan dikirimkan oleh pengguna. Proses pengirim data dari telepon seluler pengguna ke *server* menggunakan *webservice*. Data yang diterima kemudian disimpan dalam *database* untuk selanjutnya diproses menggunakan algoritma *K-Nearest Neighbors*. *Output* yang dihasilkan berupa koordinat posisi pengguna.

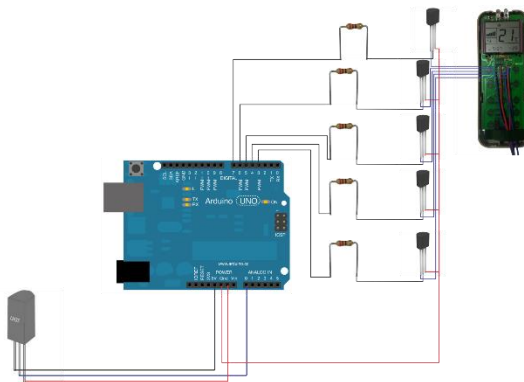
Setelah data koordinat didapatkan, tahap selanjutnya adalah melakukan estimasi jumlah orang yang terdapat di dalam ruangan. Apabila di dalam ruangan terdapat seseorang ataupun lebih maka *server* akan mengirimkan perintah kepada mikrokontroler Arduino untuk menghidupkan pendingin ruangan, dan jika pendingin ruangan dalam keadaan menyala namun tidak terdapat orang maka *server* akan memerintahkan mikrokontroler Arduino untuk mematikan pendingin ruangan.

Suhu dalam ruangan bersifat adatif terkait dengan jumlah orang yang berada di dalam ruangan dan nilai yang didapat pada sensor suhu. Apabila nilai yang didapat dari sensor suhu yang terhubung langsung pada Arduino ini melebihi nilai 28 derajat Celsius maka suhu akan diturunkan. Pada komunikasi mikrokontroler Arduino dan *server* terdapat waktu *delay* dalam pengiriman perintah maupun data yang didapat dari sensor suhu. Perubahan data ataupun data baru disimpan dalam *database* pada *server* yang berfungsi menampilkan informasi pada aplikasi web dan aplikasi yang terdapat pada telepon seluler pengguna. Aplikasi tersebut yang disebut sebagai *end-user* pada sistem ini.

3.3. Perancangan Perangkat Keras

Pada sistem ini, perangkat keras yang digunakan sebagai berikut:

- Satu buah mikrokontroler Arduino Uno rev 3,
- Satu buah Wi-Fi Shield Arduino,
- Satu buah sensor suhu LM35,
- Tiga buah *access point*,
- Satu buah pengendali jarak jauh pendingin ruangan universal,
- Satu set kabel *jumper*,
- Satu buah kabel USB,
- Satu meter kabel kecil,
- Satu buah *breadboard*,
- Lima buah resistor 22K,
- Lima buah transistor NPN,
- Satu buah laptop Asus N46VZ Inter i7 2.4 GHz dengan RAM 8.00 GB DDR3.



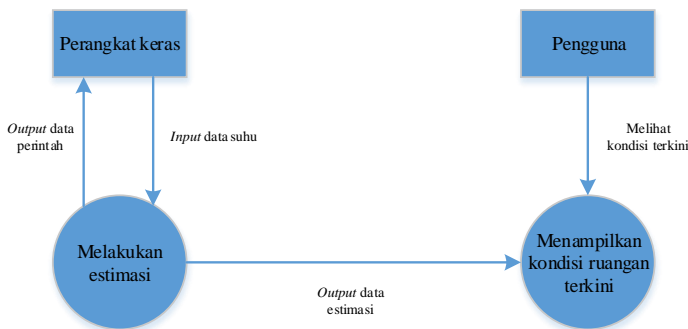
Gambar 3.2. Skema perancangan perangkat keras.

Rangkaian perangkat keras pada sistem ini ditunjukkan pada Gambar 3.2. Perangkat keras yang digunakan antara

pengendali jarak jauh pendingin ruangan dan mikrokontroler Arduino dirangkai menggunakan solder untuk menghubungkan kabel *jumper* pada pengendali jarak jauh. Komponen berupa transistor dan resistor berguna mencegah kerusakan pada mikrokontroler. Selain itu terdapat sensor suhu yang berguna mendapatkan nilai suhu yang diperlukan. Pengiriman dan penerimaan data pada mikrokontroler Arduino dikirimkan dengan Wi-Fi Shield yang berguna menghubungkan mikrokontroler kepada *access point*.

3.4. Perancangan Diagram Aliran Data Level 0

Diagram aliran data level 0 merupakan gambaran mengenai fungsionalitas sebuah sistem beserta aktor yang terlibat. Sistem yang akan dibangun dapat melakukan estimasi suhu ideal pada sebuah ruangan. Berdasarkan diagram alir data pada Gambar 3.3, sistem ini diawali dengan memasukkan data posisi pengguna yang digunakan untuk melakukan proses estimasi jumlah orang di dalam ruangan. Setelah didapatkan jumlah orang, maka selanjutnya akan dilakukan proses estimasi suhu ideal dengan masukan berupa data suhu.



Gambar 3.3. Perancangan diagram alir data level 0.

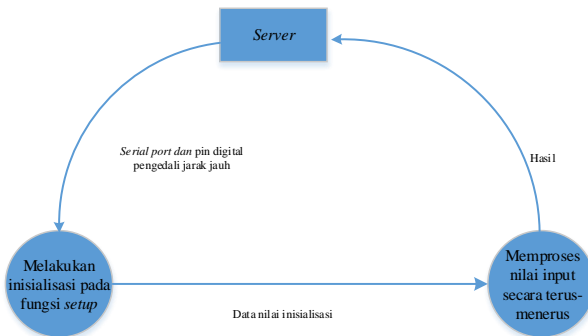
Hasil dari proses ini adalah melakukan pengiriman data estimasi suhu pada perangkat keras. Segala proses yang ada berjalan di sisi *server* dengan penggunaan *database* untuk penyimpanan data.

3.5. Perancangan Alir Aplikasi Sistem

Alur pada setiap proses yang terdapat dalam sistem digambarkan menggunakan diagram alir data untuk memudahkan pemahaman secara garis besar proses yang terdapat di dalam sistem. Diagram aliran data aplikasi sistem terdiri dari inisialisasi mikrokontroler Arduino, mendeteksi besaran nilai sensor suhu, menentukan posisi pengguna, melakukan perintah pengendali jarak jauh, menentukan jumlah pengguna, menentukan kondisi pendingin ruangan, menentukan suhu ideal secara adaptif, menjalankan aplikasi web, dan menampilkan data informasi.

3.5.1. Diagram Aliran Data Inisialisasi Mikrokontroler Arduino

Pada saat Arduino menyala, yang pertama kali dilakukan adalah melakukan inisialisasi pada fungsi *setup*. Inisialisasi hanya dilakukan sekali pada saat awal program dijalankan.

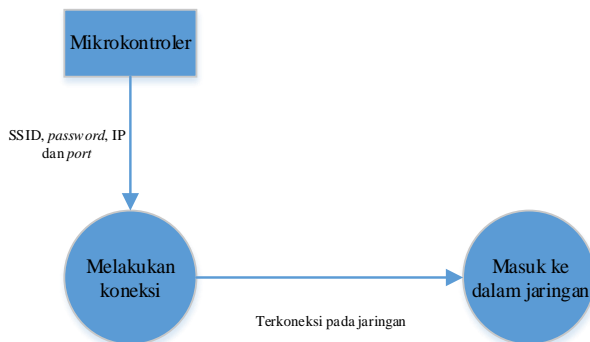


Gambar 3.4. Diagram aliran data inisialisasi mikrokontroler Arduino.

Proses ini mengatur pengaturan *serial port* dan inisialisasi digital *input* atau *ouput pin* yang akan digunakan pada proses selanjutnya. Diagram alir inisialisasi Arduino dijelaskan pada Gambar 3.4.

3.5.2. Diagram Aliran Data Melakukan koneksi WLAN pada Mikrokontroler Arduino

Setelah melakukan inisialisasi pada awal program dinyalakan. Selanjutnya merupakan tahap mikrokontroler Arduino melakukan koneksi pada *server* melalui jaringan WLAN. Tahap ini memerlukan inisialisasi nama jaringan WLAN, kata kunci, IP dan *port* yang akan digunakan. Diagram alir melakukan koneksi WLAN pada mikrokontroler Arduino dapat dilihat pada Gambar 3.5.

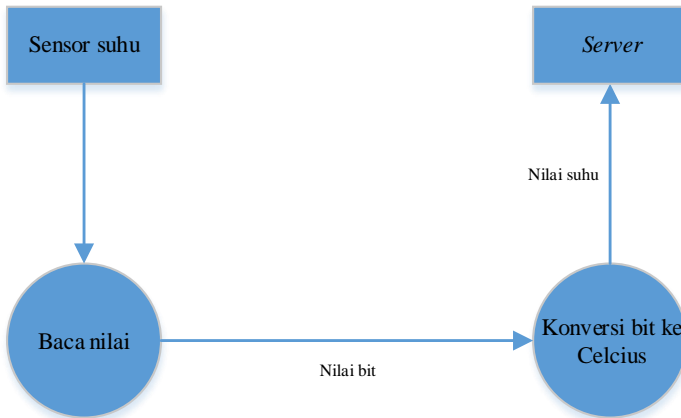


Gambar 3.5. Diagram aliran data melakukan koneksi WLAN pada mikrokontroler Arduino.

3.5.3. Diagram Aliran Data Mendeteksi Besaran Nilai Sensor Suhu (LM35)

Sensor suhu terhubung pada mikrokontroler Arduino dengan menggunakan *pin* analog dengan nilai satuan data berupa bit, dimana harus dikonversikan terlebih dahulu sebelum

mendapatkan nilai dalam satuan derajat Celcius. Inisialisasi *pin* analog pada proses sebelumnya diperlukan untuk memperoleh data dari *pin* yang terhubung antara sensor dan mikrokontroler. Dalam mendeteksi besaran nilai suhu dapat diberikan waktu *delay* atau jeda dengan lama waktu tertentu. Diagram alir untuk mendeteksi besaran nilai sensor suhu dapat dilihat pada Gambar 3.6.

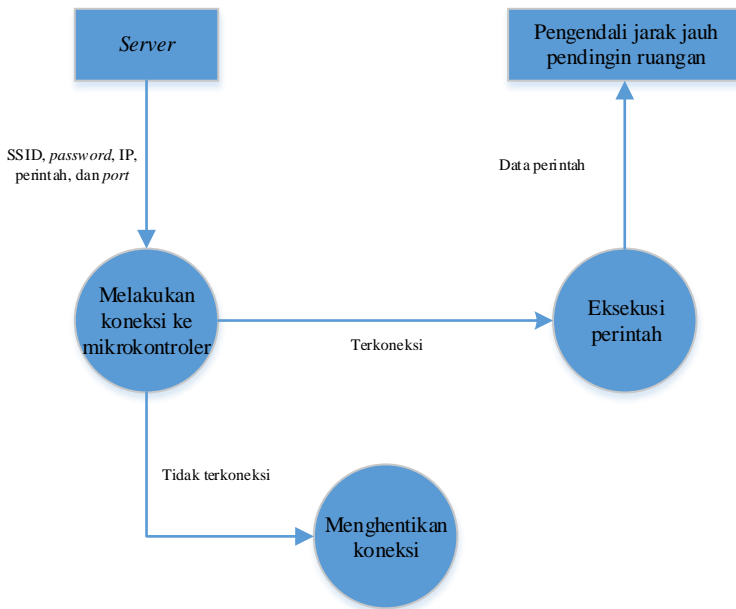


Gambar 3.6. Diagram aliran data mendeteksi besaran nilai sensor suhu.

3.5.4. Diagram Aliran Data Melakukan Perintah Pengendali Jarak Jauh

Data perintah yang dikirim dari *server* pada pengendali jarak jauh memiliki nilai berupa karakter. Namun sebelum data dikirimkan terlebih dahulu dilakukan proses koneksi terhadap mikrokontroler menggunakan *access point*, dimana terjadi inisialisasi *IP server* dan *port* yang akan digunakan dalam proses pengiriman data. Apabila keduanya telah terkoneksi pada jaringan yang sama, maka pengiriman data dapat dilakukan. Ketika data sampai pada mikrokontroler maka data langsung dikonversi untuk melakukan eksekusi perintah. Pengiriman data dapat dilakukan

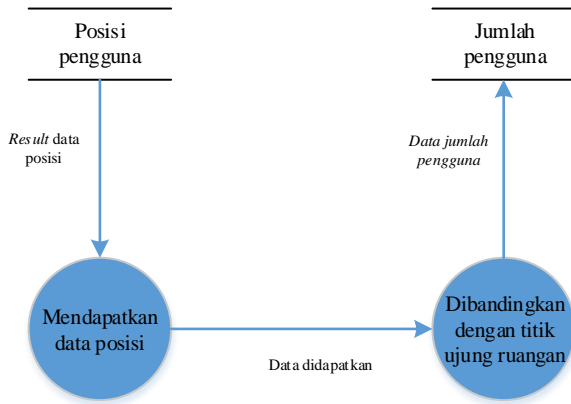
berulang kali dengan *delay* waktu tertentu. Diagram alir untuk melakukan perintah kepada pengendali jarak jauh dapat dilihat pada Gambar 3.7.



Gambar 3.7. Diagram aliran data melakukan perintah pengendali jarak jauh.

3.5.5. Diagram Aliran Data Menentukan Jumlah Pengguna

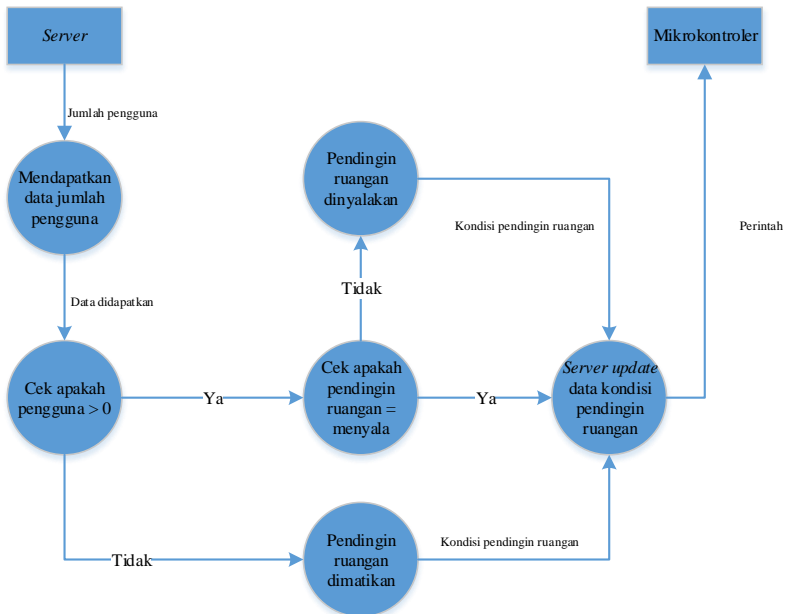
Dalam menentukan jumlah pengguna, terlebih dahulu didapatkan data letak posisi setiap pengguna pada *database*. Data tersebut kemudian dibandingkan dengan setiap titik ujung ruangan untuk didapatkan hasil apakah pengguna berada di dalam ruangan atau tidak. Kemudian data yang didapat disimpan dalam *database*, proses ini berada di sisi *server*. Diagram alir menentukan jumlah pengguna dapat dilihat pada Gambar 3.8.



Gambar 3.8. Diagram aliran data menentukan jumlah pengguna.

3.5.6. Diagram Aliran Data Menentukan Kondisi Pendingin Ruangan

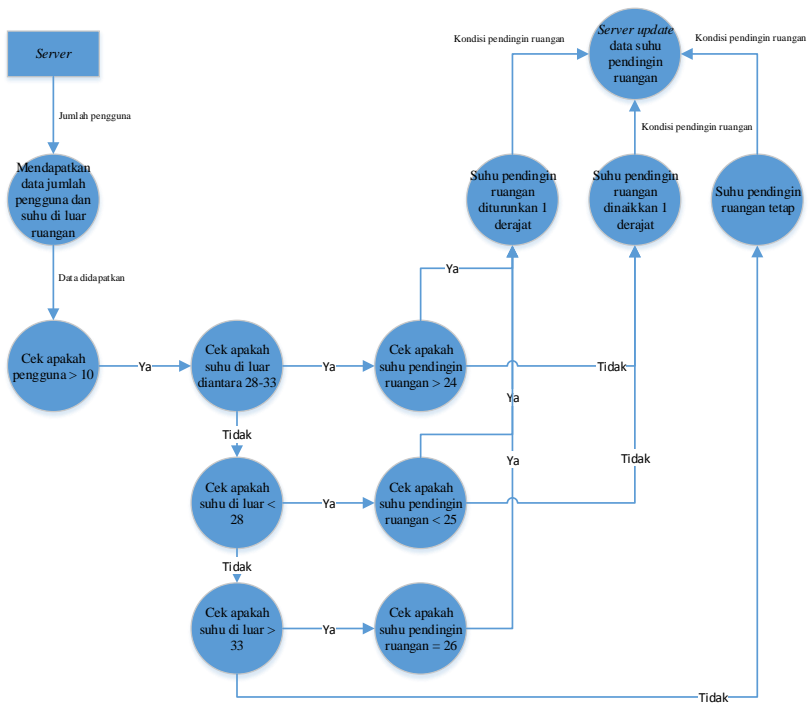
Setelah melakukan proses menentukan jumlah pengguna dalam selang waktu tertentu, proses berikutnya adalah menentukan kondisi pendingin ruangan dengan data jumlah pengguna. Proses yang terjadi adalah melakukan pengecekan jumlah pengguna yang terdapat di *database*, data tersebut menjadi bahan pertimbangan dalam menentukan kondisi pendingin ruangan. Selanjutnya data hasil penentuan kondisi akan disimpan dalam *database* dan dikirimkan ke mikrokontroler sebagai perintah. Proses ini dapat dilakukan berulang kali dengan *delay* waktu tertentu. Untuk lebih jelasnya, dapat dilihat pada Gambar 3.9.



Gambar 3.9. Diagram aliran data menentukan kondisi pendingin ruangan.

3.5.7. Diagram Aliran Data Menentukan Suhu Ideal secara Adaptif

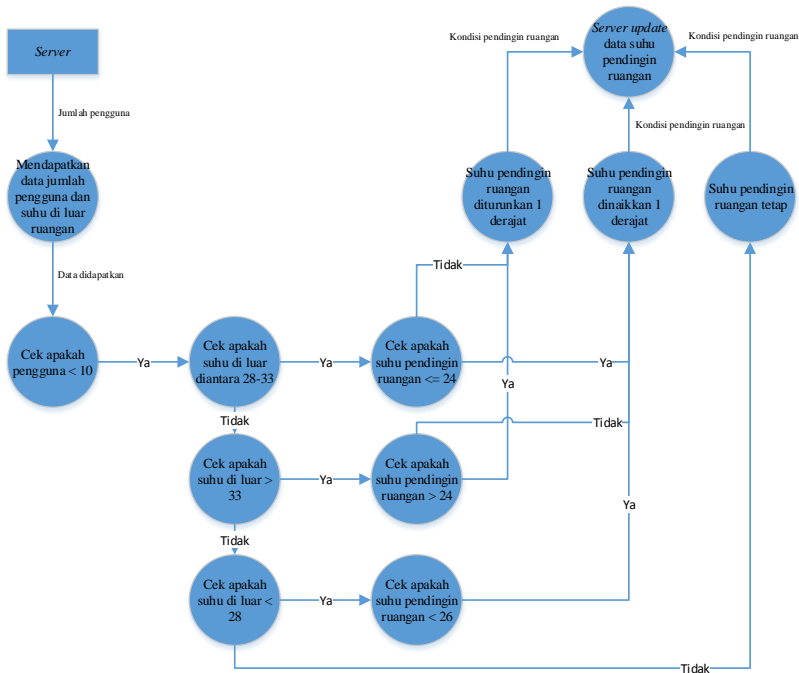
Tahap selanjutnya yaitu menentukan suhu ideal secara adaptif. Data jumlah pengguna yang didapat dalam proses sebelumnya digunakan dalam menentukan suhu ideal. Data suhu diinisialisasikan dengan nilai 25 derajat Celcius. Data nilai suhu juga ditentukan oleh data nilai suhu yang didapatkan. Diagram alir untuk menentukan suhu ideal secara adaptif ditunjukkan pada Gambar 3.10 dan Gambar 3.11.



Gambar 3.10. Diagram aliran data menentukan suhu ideal secara adaptif (Bagian 1).

Data jumlah pengguna dan data nilai suhu di luar ruangan yang didapatkan oleh *server* mempengaruhi dalam menentukan suhu ideal secara adaptif. Kedua diagram tersebut mempunyai bentuk yang sama. Namun, setelah mendapatkan data yang dibutuhkan, *server* memeriksa apakah jumlah pengguna lebih dari sepuluh pengguna. Hal ini berfungsi untuk menentukan suhu yang ideal, karena terdapat perbedaan dalam penentuan suhu ideal dengan jumlah pengguna lebih dari sepuluh pengguna dan kurang dari sepuluh pengguna. Perbedaan yang terdapat yaitu, selisih satu nilai suhu. Jika terdapat lebih dari sepuluh pengguna di dalam ruangan dan suhu di luar di antara 28° hingga 33° Celcius, maka

suhu ideal yang di tentukan sebesar 24° Celcius. Sebaliknya, apabila terdapat pengguna dalam jumlah kurang dari sebelas pengguna, maka suhu ideal yang ditetapkan adalah 25° derajat Celcius. Suhu pada pendingin ruangan berubah secara adaptif mengikuti pertambahan atau pengurangan jumlah pengguna di dalam ruangan dan kenaikan atau turunnya data nilai suhu yang didapatkan oleh *server*.

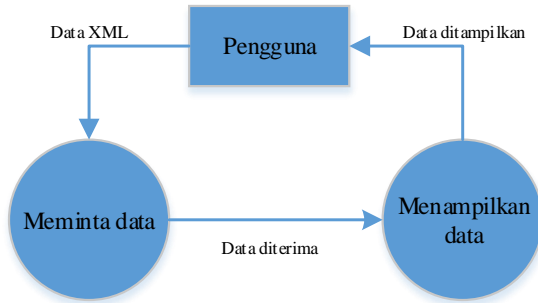


Gambar 3.11. Diagram aliran data menentukan suhu ideal secara adaptif (Bagian 2).

3.5.8. Diagram Aliran Data Menjalankan Aplikasi Web

Aplikasi web menampilkan data informasi kondisi pendingin ruangan pada sebuah halaman web. Terdapat *map*

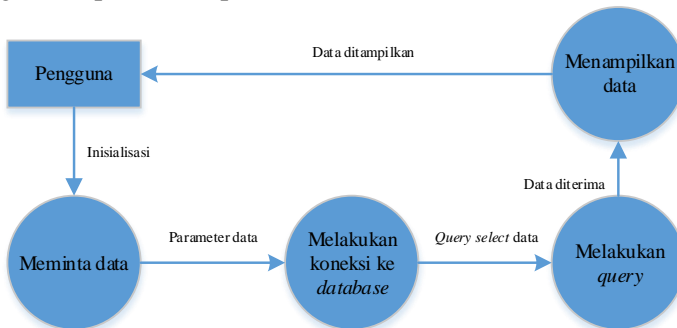
dengan sebuah titik yang menggambarkan letak pengguna. Diagram alir menjalankan aplikasi web dapat dilihat pada Gambar 3.12.



Gambar 3.12. Diagram alir data menjalankan aplikasi web.

3.5.9. Diagram Aliran Data Menampilkan Data Informasi pada *Webservice*

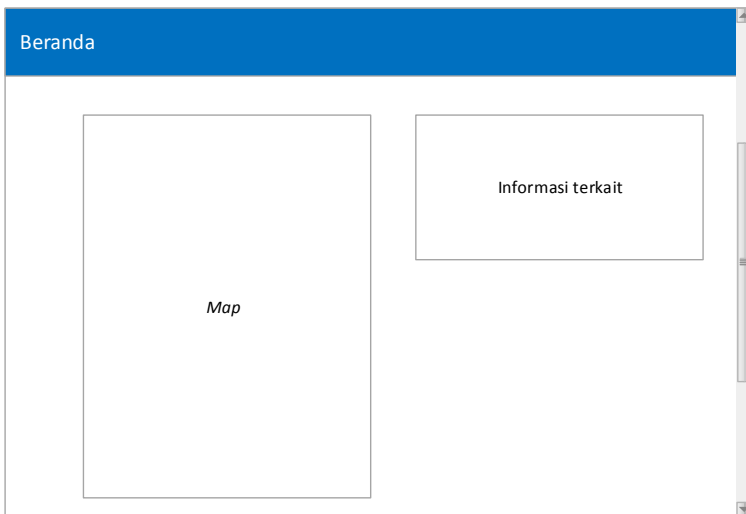
Proses menampilkan data informasi ini dimulai dari melakukan *query select* pada *database*. Proses ini berguna untuk menampilkan data informasi sistem terkini, yaitu: menampilkan informasi terkait dengan kondisi pendingin ruangan, nilai suhu, dalam jumlah orang. Kemudian data ditampilkan pada sebuah web. Diagram dapat dilihat pada Gambar 3.13.



Gambar 3.13. Diagram alir data menampilkan data informasi.

3.6. Rancangan Antarmuka Sistem

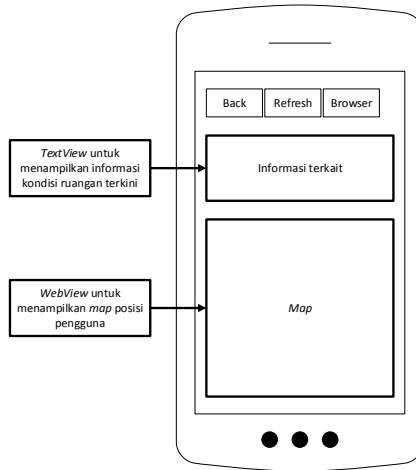
Rancang antarmuka yang dibangun untuk melihat informasi data dari sistem terdapat pada aplikasi web dan aplikasi *mobile*. Terdapat *map* dan *panel* yang berisikan informasi terkini pada sistem. *Map* ditampilkan untuk melihat titik lokasi dimana pengguna berada. Dimana berisikan denah dari sebuah ruangan yang terdapat satu titik ataupun lebih bergantung dengan jumlah pengguna yang menandakan dimana letak pengguna berada. *Panel* berfungsi menampilkan informasi mengenai sistem, seperti: jumlah orang yang berada di dalam ruangan, kondisi mengenai pendingin ruangan, suhu pendingin ruangan, dan suhu yang terdapat di luar ruangan.



Gambar 3.14. Rancangan antarmuka sistem pada aplikasi web.

Rancang bangun antarmuka pada aplikasi web dari sistem ini dapat dilihat pada Gambar 3.14. Ketika pengguna membuka aplikasi web maka akan tampil halaman web tersebut. Dimana

pengguna dapat mengetahui informasi yang ditampilkan pada sistem yang dibangun.



Gambar 3.15. Rancangan antarmuka sistem pada aplikasi *mobile*.

Pengguna juga dapat melihat informasi menggunakan aplikasi *mobile* pada ponsel *pintar*. Rancangan antarmuka sistem pada aplikasi *mobile* menampilkan informasi yang sama dengan informasi yang ditampilkan oleh rancangan antarmuka sistem pada aplikasi web. Dimana pengguna dapat melihat informasi kondisi ruangan terkini dan *map* yang berisikan titik letak pengguna. Gambar 3.15 merupakan rancang bangun antarmuka sistem pada aplikasi *mobile*.

BAB IV IMPLEMENTASI

Bab ini akan membahas mengenai implementasi sistem pada perangkat keras dan perangkat lunak yang meliputi tahap implementasi pada program berupa *pseudocode* dari perancangan sistem yang telah dibahas pada bab 3.

4.1. Lingkungan Implementasi

Pada tahap merancang dan melakukan implementasi perangkat lunak ini digunakan beberapa perangkat pendukung sebagai berikut.

4.1.1. Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan pada lingkungan pengembangan sistem adalah komputer dan mikrokontroler Arduino. Spesifikasi perangkat keras tersebut adalah sebagai berikut:

- Laptop ASUS N46vz
 - Windows 8.1 Pro 64 bit,
 - Prosesor Intel® Core(TM) i7-3630QM CPU @ 2.40 GHz, dan
 - RAM 8,00 GB DDR3.
- Arduino Uno
 - ATmega328P dengan 32 KB flash memori,
 - 2 KB memori SRAM,
 - 1 KB memori EEPROM, dan
 - 16 MHz *clock speed*.

4.1.2. Lingkungan Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pengembangan sistem adalah sebagai berikut:

- Microsoft Windows 8.1 Pro sebagai sistem operasi,
- Eclipse Juno v4.2.2 sebagai IDE untuk mengimplementasikan aplikasi *server*,
- Android Development Tools v22.6.2 untuk mengimplementasikan aplikasi *mobile*,
- Arduino Development Kit v1.0.2 sebagai IDE untuk mengimplemtasikan perangkat lunak pada mikrokontroler Arduino,
- MySQL 5.6.16 sebagai *database server*,
- Apache 2.4.7 sebagai *platform web server* untuk menjalankan aplikasi web,
- Google Chrome sebagai *browser* dalam tahap uji coba, dan
- Microsoft Visio 2013 untuk merancang diagram alir data sistem.

4.2. Implementasi Perangkat Keras

Pada bab ini akan dibahas mengenai pembuatan dari implementasi perangkat keras. Tahap ini diawali dengan membuat sebuah *prototype* untuk melakukan uji coba apakah perangkat keras dapat berfungsi dalam mengambil data pada sensor, menerima data dari *server*, dan mengirim data ke *server*. Perangkat keras yang digunakan pada sistem ini sebagai berikut:

- Satu buah mikrokontroler Arduino Uno rev 3,
- Satu buah Wi-Fi Shield Arduino,
- Satu buah sensor suhu LM35,
- Tiga buah *access point*,
- Satu buah pengendali jarak jauh pendingin ruangan universal,
- Satu set kabel *jumper*,
- Satu buah kabel USB,
- Satu meter kabel kecil,

- Satu buah *breadboard*,
- Lima buah resistor 22K,
- Lima buah transistor NPN, dan
- Satu buah laptop Asus.

Tugas Akhir ini menggunakan *prototype* perangkat keras berupa mikrokontroler Arduino yang dilengkapi dengan sebuah sensor suhu (LM35), Arduino Wi-Fi shield, dan kabel *jumper* tersambung pada pengendali pendingin ruangan jarak jauh. Mikrokontroler berfungsi sebagai perangkat utama untuk melakukan kendali perintah terhadap pendingin ruangan.

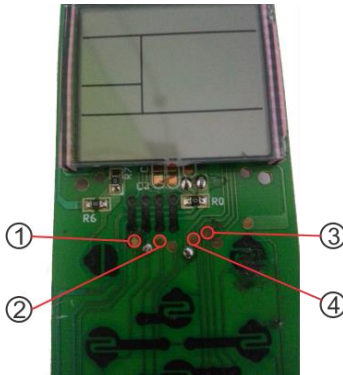
Wi-Fi shield disambungkan ke Arduino Uno dengan cara memasukkan *pin* bagian bawah Wi-Fi shield pada *pin input/output* dari Arduino Uno. *Breadboard* berfungsi sebagai penghubung antara mikrokontroler dengan sensor suhu. Sensor suhu menancap pada bagian atas *breadboard* melalui kabel *jumper*, memiliki tiga *pin* utama, yaitu *pin vcc*, *ground*, dan *output* yang berupa *pin* analog.

Untuk menghubungkan pengendali pendingin ruangan jarak jauh dengan mikrokontroler Arduino, terlebih dahulu harus membuka penutup depan dan penutup belakang. Apabila penutup sudah terbuka, maka pengendali pendingin ruangan jarak jauh akan tampak seperti pada Gambar 4.1.



Gambar 4.1. Pengendali pendingin ruangan jarak jauh yang sudah dibuka penutupnya.

Tahap selanjutnya adalah melakukan proses penyolderan kabel pada pengendali pendingin ruangan jarak jauh untuk menghubungkan komponen perangkat. Bagian yang dilakukan penyolderan, yaitu bagian lingkaran tembaga. Bagian tersebut dapat dilihat pada Gambar 4.2. Hal ini bergantung dengan tipe ataupun merek pengendali pendingin ruangan jarak jauh.



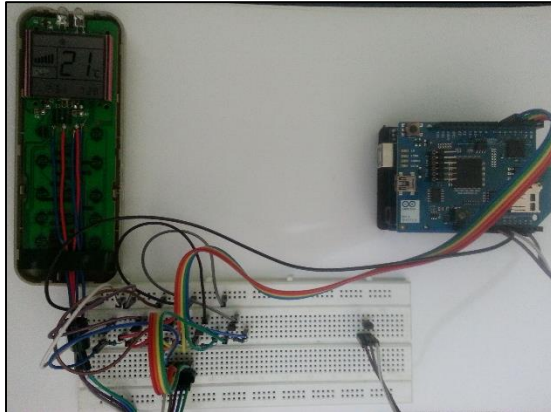
Gambar 4.2. Bulatan tembaga pada pengendali pendingin ruangan jarak jauh.

Apabila dua bagian lingkaran tembaga disambungkan menggunakan kabel maka akan memberikan perintah terhadap pengendali pendingin ruangan jarak jauh, seperti:

- Perintah menghidupkan atau mematikan pendingin ruangan diberikan apabila lingkaran tembaga nomor satu dan nomor dua dihubungkan.
- Perintah menaikkan suhu pendingin ruangan terjadi apabila lingkaran tembaga nomor dua dan nomor tiga dihubungkan.
- Perintah untuk menurunkan suhu terjadi apabila lingkaran tembaga nomor dua dan nomor empat saling dihubungkan.

Pengendali pendingin ruangan jarak jauh yang sudah terhubung dengan kabel yang tertancap pada *breadboard*,

kemudian disambungkan pada resistor dan transistor seperti pada skema yang sudah dijelaskan dalam bab sebelumnya. Gambar 4.3 merupakan gambar dari rangkaian dari komponen perangkat keras yang sudah dihubungkan.



Gambar 4.3. Rangkaian perangkat keras.

4.3. Implementasi Perangkat Lunak

Implementasi pada perangkat lunak terbagi menjadi empat bagian yaitu mikrokontroler Arduino, *server*, aplikasi web dan aplikasi *mobile*. Setiap bagian dari implementasi akan dijelaskan lebih lanjut pada setiap subbab.

4.3.1. Implementasi pada Mikrokontroler Arduino

- Inisialisasi mikrokontroler Arduino

Proses inisialisasi dilakukan hanya pada saat awal mikrokontroler dinyalakan. Proses ini merupakan hal yang penting dalam penggunaan *pin* pada Arduino. Dikarenakan untuk menggunakan *pin* tersebut terlebih dahulu dilakukan inisialisasi *pin* yang akan digunakan dalam proses pembuatan perangkat

lunak. Inisialisasi Arduino pada Gambar 4.4 dapat dijelaskan dalam beberapa tahapan, yaitu:

1. Buka *serial port* dan inisialisasi *baud rate*.
2. Inisialisasi *pin mode* pengendali jarak jauh.
3. Inisialisasi digital *write* untuk pengendali jarak jauh.

1	Open Serial.begin(baud rate)
2	Initialize <i>pinMode</i> (remote <i>pin</i> , OUTPUT)
3	Initialize digitalMode(remote <i>pin</i> , LOW)

Gambar 4.4. Inisialisasi Arduino *serial port* dan *pin* pengendali pendingin ruangan jarak jauh.

- Melakukan koneksi terhadap *server*

Setelah melakukan inisialisasi *serial port*, *pin mode*, dan digital *write* pada implementasi fungsi *setup* terdapat pula inisialisasi koneksi terhadap *server*. Hal ini berfungsi menghubungkan mikrokontroler Arduino dengan jaringan *wireless*. Sebelum melakukan proses ini, yang perlu dilakukan adalah deklarasi *variable* nama jaringan WLAN, kata kunci, IP, dan *port* yang akan digunakan oleh *server*. Gambar 4.5 merupakan tahapan mikrokontroler Arduino melakukan koneksi terhadap *server* yang dapat dijelaskan sebagai berikut:

1. Buka pustaka Wi-Fi dan inisialisasi SSID serta *password*.
2. Koneksi pada *server*.
3. Jika mikrokontroler tidak dapat melakukan koneksi maka aplikasi akan mengirim pesan gagal tanpa mengirimkan pesan berhasil dan mendeteksi nilai suhu.

Input: SSID, password, IP, dan port	
1	Open WiFi.begin(SSID,password)
2	Initialize client.connect(IP,port)
3	If client connected
4	Then
5	Print "Connected"
6	Call function HitungSuhu
7	Else

8	Then
9	Print "Connction failed"

Gambar 4.5. Mikrokontroler melakukan koneksi WLAN.

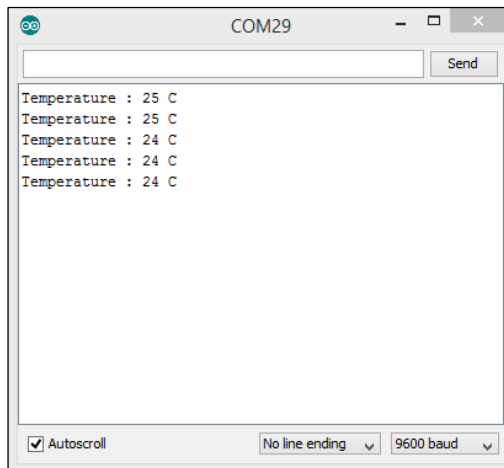
- Mendeteksi Besaran Nilai Sensor Suhu (LM35)

Input yang digunakan untuk mendeteksi besaran nilai sensor suhu merupakan analog *input*. Oleh karena itu perlu dilakukan konversi untuk mendapatkan nilai suhu dalam satuan derajat Celcius. Implementasi untuk mendeteksi besaran nilai sensor suhu dapat dilihat pada Gambar 4.6 dan dijelaskan dalam beberapa tahap sebagai berikut:

1. Mengambil nilai *pin* analog.
2. Konversi nilai dalam satuan besaran Celcius.

1	set var1 to analogRead(temperature <i>pin</i>)
2	set val to (5.0 * var1 *100.0) / 1024
Output : nilai suhu dalam satuan besaran Celcius	

Gambar 4.6. Implementasi mendeteksi besaran nilai suhu.



Gambar 4.7. Screenshot deteksi besaran nilai suhu.

Gambar 4.7 menunjukkan hasil *screenshot* dari implementasi mendeteksi besaran nilai suhu pada sensor suhu.

- Melakukan Perintah Pengendali Jarak Jauh

Implementasi dapat melakukan perintah terhadap pengendali jarak jauh, apabila terdapat data yang dikirimkan oleh *server* berupa karakter. Setiap karakter yang diterima oleh mikrokontroler menginisialkan masing-masing perintah. Pada Gambar 4.8 menunjukkan tahap dalam melakukan perintah pengendali jarak jauh yang dapat dijelaskan sebagai berikut:

1. Jika klien tidak tersambung pada *server*, maka klien akan menghentikan koneksi.
2. Mengambil data yang dikirimkan oleh *server*.
3. Jika data yang dikirimkan berupa karakter 'H' atau 'M', maka diberikan perintah untuk menghidupan/mematikan.
4. Jika data berupa karakter 'X', perintah yang diberikan adalah menaikkan suhu.
5. Jika data yang dikirimkan berupa karakter 'Y', maka perintah yang diberikan adalah menurunkan suhu.
6. Melakukan koneksi kepada *server*.

```

1  if client.avaiaible
2      then
3          set char to client.read
4          if char == H
5              then
6                  call function Power
7          if char == X
8              then
9                  call function SuhuNaik
10         If char == Y
11             then
12                 call function SuhuTurun
13     else
14         then
15         Client.stop

```

16	set delay to 10000 //miliseconds
17	call function Connection

Gambar 4.8. Implementasi melakukan perintah pengendali jarak jauh.

Dalam memberikan perintah pengendali jarak jauh terdapat proses penulisan nilai pada *pin* digital yang sudah diinisialisasi sebelumnya. Gambar 4.9 menjelaskan bagaimana mikrokontroler Arduino melakukan proses penulisan nilai digital untuk dapat memberikan perintah pada pengendali pendingin ruangan jarak jauh, sebagai berikut:

1. Program mengubah nilai digital pada *pin* digital menjadi *HIGH*.
2. Melakukan *delay* pengambilan selama 1000 *miliseconds*.
3. Mengubah nilai digital pada *pin* menjadi *LOW*.

1	set digitalWrite(remote <i>pin</i> ,HIGH)
2	set delay to 1000 //miliseconds
3	set digitalWrite(remote <i>pin</i> ,LOW)

Gambar 4.9. Implementasi penulisan nilai pada *pin* digital.

4.3.2. Implementasi pada *Server*

- Menentukan Jumlah Pengguna

Program dapat menentukan jumlah pengguna, berdasarkan dengan data yang dihasilkan oleh *Indoor Positioning System*. Data yang didapatkan dari *database* berupa nilai *x* dan *y* letak posisi dimana pengguna berada. Dengan adanya data ini maka dapat dilakukan estimasi jumlah pengguna yang berada di ruangan. Sebelum melakukan proses ini, perlu dilakukan pengukuran terhadap ruangan yang digunakan untuk mendapatkan nilai titik ujung. Proses ini membandingkan nilai *x* dan *y* posisi pengguna dengan nilai setiap titik ujung ruangan. Implementasi ini berjalan pada *server* dan *webservice*. Pada Gambar 4.10 ditunjukkan

tahapan untuk menentukan jumlah pengguna dengan penjelasan sebagai berikut:

1. Melakukan koneksi ke *database*.
2. Mengambil nilai pada *database*.
3. Jika nilai data lebih besar dari titik ujung depan ruangan, maka pengguna diruangan bertambah.

1	open connection to database
2	set var1 to db-> query(select*from table)
3	set x to var1(x_user)
4	set y to var1(y_user)
5	if y > 91.0
6	then
7	set pengguna to pengguna + 1

Gambar 4.10. Implementasi menentukan jumlah pengguna.

- Menentukan Kondisi Pendingin Ruangan

Bagian ini menjelaskan bagaimana *server* dapat menentukan kondisi pendingin ruangan, berdasarkan data jumlah pengguna yang sudah didapatkan pada tahap sebelumnya. Pendingin ruangan dinyalakan apabila terdapat pengguna dalam ruangan, dan sebaliknya pendingin ruangan akan dimatikan apabila tidak terdapat pengguna. Gambar 4.11 Merupakan cara *server* untuk menentukan kondisi pendingin ruangan dengan penjelasan sebagai berikut:

1. Mendapatkan data jumlah pengguna.
2. Melakukan pengecekan, jika jumlah pengguna melebihi satu, maka pendingin ruangan akan dinyalakan. Jika jumlah kurang dari satu maka pendingin ruangan dimatikan.
3. Menyimpan data kondisi pendingin ruangan terkini.
4. Melakukan pengiriman data pada mikrokontoler.

1	set var1 to jumlahPengguna
2	if var1 > 0
3	then

4	Set PowerAc to on
5	else
6	then
7	set PowerAc to off
8	insert data to database
9	send result to microcontroller

Gambar 4.11. Implementasi menentukan kondisi pendingin ruangan.

- **Menentukan Suhu Ideal secara Adaptif**

Setelah *server* menentukan kondisi pendingin ruangan, selanjutnya adalah menentukan suhu ideal secara adaptif. Nilai suhu yang telah dideteksi oleh mikrokontroler menggunakan sensor berfungsi untuk menentukan suhu yang ideal. Implementasi untuk menentukan suhu ideal secara adaptif ditunjukkan pada Gambar 4.12 dan dijelaskan sebagai berikut:

1. Mendapatkan data jumlah pengguna.
2. Inisialisasi nilai suhu ideal ruangan.
3. Mendapatkan data nilai suhu pada sensor.
4. Mendapatkan nilai suhu pada pendingin ruangan.
5. Jika jumlah pengguna lebih dari sepuluh pengguna, dilakukan pengecekan terhadap nilai suhu pada sensor dan suhu pada pendingin ruangan.
6. Jika suhu di luar ruangan memiliki rentang nilai antara 28 hingga 33, maka dilakukan pengecekan terhadap suhu pendingin ruangan. Apabila suhu pendingin ruangan lebih besar dari 24 maka nilai suhu dikurangi satu dan sebaliknya jika tidak suhu pendingin ruangan ditambahkan satu.
7. Jika data nilai suhu di luar ruangan lebih kecil dari 28, maka dilakukan pengecekan terhadap suhu pendingin ruangan. Apabila suhu pendingin ruangan lebih besar dari 24, maka suhu pendingin ruangan dikurangi satu dan jika tidak suhu ditambahkan satu.

8. Jika suhu di luar ruangan lebih dari 33 dan suhu pada pendingin ruangan sama dengan 26 maka suhu pendingin ruangan dikurangi satu.
9. Apabila jumlah pengguna kurang dari sepuluh pengguna, maka dilakukan pengecekan terhadap suhu pendingin ruangan dan suhu di luar ruangan.
10. Jika suhu di luar ruangan memiliki rentang nilai antara 28 hingga 33, maka dilakukan pengecekan terhadap suhu pendingin ruangan. Apabila suhu pendingin ruangan lebih kecil dari 24 maka nilai suhu ditambah satu dan sebaliknya jika tidak suhu pendingin ruangan dikurangi satu.
11. Jika data nilai suhu di luar ruangan lebih besar dari 33, maka dilakukan pengecekan terhadap suhu pendingin ruangan. Apabila suhu pendingin ruangan lebih besar dari 24, maka suhu pendingin ruangan dikurangi satu dan jika tidak suhu ditambahkan satu.
12. Jika suhu di luar ruangan lebih besar dari 28 dan suhu pada pendingin ruangan lebih kecil dari 26 maka suhu pendingin ruangan ditambah satu.
13. Melakukan penyimpanan data pada *database*.

```

1  set var1 to jumlahPengguna
2  set suhu to 25
3  set var2 to SuhuSensor
4  set var3 to SuhuPendinginRuangan
5  if var1 > 10
6    then
7      If var2 >= 28 and var2 <= 33
8      Then
9        If var3 > 24
10       Then
11         set suhu to suhu - 1
12       Else
13       Then
14         set suhu to suhu + 1
15     If var2 < 28
16     Then
17       If var3 < 25

```

```

18         Then
19             set suhu to suhu - 1
20         Else
21             Then
22                 set suhu to suhu + 1
23         If var2 > 33
24             Then
25                 If var3 = 26
26                     Then
27                         set suhu to suhu - 1
28         if var1 < 10
29             then
30         If var2 >= 28 and var2 <= 33
31             Then
32                 If var3 <= 24
33                     Then
34                         set suhu to suhu + 1
35                     Else
36                         Then
37                             set suhu to suhu - 1
38         If var2 > 33
39             Then
40                 If var3 > 24
41                     Then
42                         set suhu to suhu - 1
43                     Else
44                         Then
45                             set suhu to suhu + 1
46         If var2 < 28
47             Then
48                 If var3 < 26
49                     Then
50                         set suhu to suhu + 1
51         insert data to database

```

Gambar 4.12. Implementasi menentukan suhu ideal secara adaptif.

- Implementasi Penerimaan dan Pengiriman Data pada Sisi *Server*

Pada bagian ini menjelaskan bagaimana sebuah *server* dan mikrokontroler Arduino melakukan penerimaan dan pengiriman

data menggunakan TCP. Sebelum memulai pertukaran data diantara keduanya, perlu dilakukan inisialisasi pada sisi *server*. Pada Gambar 4.13 ditunjukkan cara *server* melakukan penerimaan dan pengiriman data dengan penjelasan sebagai berikut:

1. Inisialisasi *socket*.
2. Menerima koneksi yang datang.
3. Inisialisasi *buffered reader*.
4. Inisialisasi data *output stream*.
5. Membaca data yang sudah dikirimkan oleh klien.
6. Mengirimkan data pada klien.

```

Input : port
1  open ServerSocket(port)
2  do ServerSocket.accept()
3  Initilize
   BufferedReader(InputStreamReader(getInputStream)
   )
4  Initialize DataOutputStream(getOutputStream)
5  set var1 to clientInput.readline
6  do clientOutput.writeBytes(data)

```

Gambar 4.13. Implementasi penerimaan dan pengiriman data pada sisi *server*.

4.3.3. Implementasi Aplikasi Web

Aplikasi web ini merupakan bentuk *end-user* pada sistem ini. Implementasi yang dilakukan pada aplikasi ini menggunakan data yang diambil dari *webservice* dengan bentuk XML.

```

1  open connection to database
2  load database
3  set query to db-> query("select*from table)
4  set header(content-type:text/xml)
5  set var1 to query
6  do str_replace(&,en,var1)

```

Gambar 4.14. Implementasi menampilkan data informasi pada *webservice*.

Implementasi untuk menampilkan data informasi pada *webservice* ditunjukkan pada Gambar 4.14. Setelah data ditampilkan dalam bentuk XML dalam *webservice*, kemudian aplikasi web membuka berkas XML pada alamat web yang sudah ditentukan untuk menampilkan data yang tersedia dalam XML pada halaman aplikasi web. Pada Gambar 4.15 merupakan implementasi pada aplikasi web dengan penjelasan sebagai berikut:

1. Inisialisasi berkas XML.
2. Mendapatkan data pada berkas XML.

1	<code>open simplexml_load_file(URL)</code>
2	<code>set var1 to xml-> result</code>
3	<code>print var1</code>

Gambar 4.15. Implementasi menjalankan aplikasi web.

4.3.4. Implementasi Aplikasi *Mobile*

Implementasi dalam aplikasi *mobile* merupakan salah satu bentuk *end-user* pada sistem ini. Aplikasi ini mengambil data pada *webservice* dalam bentuk XML untuk ditampilkan pada halaman aplikasi. Pada Gambar 4.16 ditunjukan tahapan implementasi aplikasi *mobile* dapat dijelaskan sebagai berikut:

1. Inisialisasi XML *parser*.
2. Mendapatkan berkas XML.
3. Mendapatkan data dari berkas XML.

1	<code>Initialize XMLParser</code>
2	<code>set var1 to parser.getAllPoint(URL)</code>
3	<code>set var2 to retrieve(parser,var1,key)</code>
4	<code>print var2</code>

Gambar 4.16. Implementasi aplikasi *mobile*.

4.4. Implementasi Antar Muka Perangkat Lunak

Implementasi antar muka perangkat lunak pada sistem ini terdapat pada aplikasi web dan aplikasi *mobile*. Pada saat pengguna melakukan akses terhadap aplikasi web tersebut, maka akan tampak sebuah halaman yang berisikan *map* dan informasi terkini mengenai kondisi ruangan beserta kondisi pendingin ruangan. *Map* dalam halaman web ini menggambarkan denah ruangan Lab NCC (*Net Centric Computing*) pada gedung Teknik Informatika ITS. Dalam denah tersebut terdapat titik berwarna yang menggambarkan titik letak dimana pengguna sedang berada. Gambar 4.17 merupakan *screenshot* antar muka aplikasi web pada sistem ini.



Gambar 4.17. *Screenshot* antar muka aplikasi web.

Aplikasi web dan *mobile* ini menampilkan data informasi mengenai kondisi ruangan dan kondisi pendingin ruangan, seperti data jumlah pengguna yang sedang berada di dalam ruangan, kondisi pendingin ruangan sedang menyala atau tidak, suhu pendingin ruangan, dan data nilai suhu di luar ruangan. Gambar 4.18 merupakan *screenshot* antar muka pada aplikasi *mobile*.



Gambar 4.18. Screenshot antar muka aplikasi *mobile*.

BAB V

UJI COBA DAN EVALUASI

Bab ini akan membahas uji coba dan evaluasi dari sistem yang akan dibangun. Sistem akan diuji coba secara fungsionalitas dan performa dengan melakukan skenario yang sudah ditentukan. Uji coba ini dilakukan untuk dapat mengetahui hasil daripada sistem sehingga menjawab rumusan masalah yang terdapat pada Bab I.

5.1 Lingkungan Uji Coba

Pada subbab ini menjelaskan gambaran lingkungan yang digunakan untuk melakukan uji coba aplikasi dari sistem ini. Dalam uji coba digunakan sebuah laptop sebagai *server*, sebuah mikrokontroler Arduino, sebuah ponsel *pintar* berbasis Android sebagai klien, dan pengendali pendingin ruangan jarak jauh yang diletakkan di dekat pendingin. Spesifikasi adalah sebagai berikut:

- Laptop ASUS N46VZ
 - Spesifikasi perangkat keras
 - Intel® Core™ i7-3630QM CPU @ 2.40GHz dan
 - RAM 8,00 GB
 - Spesifikasi perangkat lunak
 - Windows 8.1 Pro 64-bit sebagai sistem operasi yang digunakan,
 - Eclipse Juno v4.2.2 sebagai IDE untuk mengimplementasikan aplikasi *server*,
 - Android Development Tools v22.6.2 untuk mengimplementasikan aplikasi *mobile*,
 - Arduino Development Kit v1.0.2 sebagai IDE untuk mengimplementasikan perangkat lunak pada mikrokontroler Arduino,
 - MySQL 5.6.16 sebagai *database* yang digunakan,
 - Apache 2.4.7 sebagai web *server* untuk menjalankan aplikasi web, dan

- Google Chrome sebagai *browser* untuk menampilkan *map* dan informasi kondisi ruangan.
- Arduino Uno Rev.3
 - Spesifikasi perangkat keras
 - Atmega328 dengan 32 KB flash memori
 - 2 KB memori SRAM,
 - 1 KB memori EEPROM, dan
 - 16 MHz *clock speed*.
- Perangkat ponsel *pintar* Samsung Note 2 N7100
 - Sistem Operasi: Android v4.1.2 (Jelly Beans),
 - Chipset: Exynos 4412,
 - CPU: 1.6 GHz Cortex-A9,
 - *Memory Internal*: 16GB,
 - Speed: HSDPA, 21 Mbps, dan
 - WLAN: Wi-Fi 802.11 a/b/g/n, DLNA, Wi-Fi *hotspot*.
- Perangkat keras lainnya, seperti:
 - Pengendali pendingin ruangan *wireless* universal Chunghop K-1028E,
 - Wi-Fi Shield Arduino,
 - Sensor suhu LM35,
 - Tiga buah *access point*,
 - Satu set kabel *jumper*,
 - Kabel USB, dan
 - Satu meter kabel kecil,
 - *Breadboard*,
 - Lima buah resistor 22K, dan
 - Lima buah transistor NPN.

Uji coba sistem ini dilakukan di laboratorium NCC(*Net Centric Computing*) Teknik Informastika ITS, tampak bagian dalam pada ruang uji coba dapat dilihat Gambar 5.1. Ruang uji coba ini memiliki luas sebesar 67.62m² dengan panjang ruangan 8.75m dan lebar 7m, dan luas tersebut juga termasuk dengan teras

di depan ruang uji coba. Teras tersebut memiliki panjang 0.91m dan lebar 7m.



Gambar 5.1. Tampak dalam ruang uji coba.



Gambar 5.2. Denah lokasi uji coba.

Gambar 5.2 merupakan denah dari ruang uji coba. Pengendali pendingin ruangan jarak jauh yang tersambung dengan mikrokontroler Arduino diletakkan tidak menggunakan aturan ataupun rumus khusus dengan laptop sebagai *server*. Kedua perangkat keras tersebut akan dikoneksikan dengan *access point* yang sama untuk IPS, hal ini berfungsi untuk membuat kedua perangkat terkoneksi pada jaringan yang sama.

Untuk peletakan pengendali pendingin ruangan jarak jauh, ditempatkan dekat pendingin ruangan agar dalam pengiriman data melalui inframerah dari pengendali pendingin ruangan jarak jauh menuju pendingin ruangan dapat dikirimkan dengan baik. Apabila data inframerah yang dikirimkan pengendali tidak dapat sampai pada pendingin ruangan maka akan menyebabkan perintah tidak dapat dieksekusi.

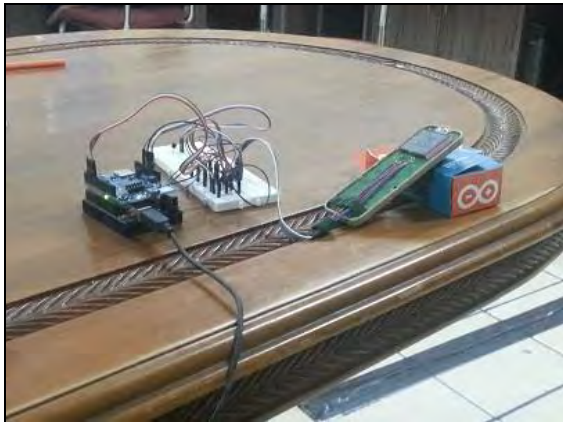


Gambar 5.3. Denah pengendali pendingin ruangan jarak jauh.

Gambar 5.3 merupakan denah ruangan dengan terdapat di dalamnya pengendali pendingin jarak jauh beserta perangkat keras

lainnya, seperti mikrokontroler Arduino dengan Wi-Fi *shield*, *breadboard*, sensor suhu, transistor, dan resistor diletakkan berdekatan dengan pendingin ruangan di dalam ruang laboratorium. Titik satu merupakan tempat mikrokontroler Arduino beserta perangkat keras lainnya diletakkan, dan titik dua merupakan tempat pendingin ruangan.

Mikrokontroler Arduino beserta perangkat keras lainnya diletakkan di atas tempat penyimpanan barang yang menghadap langsung ke arah pendingin ruangan. Untuk lebih jelasnya dapat dilihat pada Gambar 5.4.



Gambar 5.4. Posisi mikrokontroler Arduino dan perangkat keras lainnya.

5.2 Skenario Uji Coba

Uji coba dilakukan untuk melakukan pengujian terhadap fungsionalitas pada sistem yang dibangun dalam mengetahui apakah fungsionalitas utama yang terdapat dalam tahap perancangan maupun implementasi sebelumnya benar-benar diimplementasikan dan bekerja seperti seharusnya. Tahap ini didasarkan pada beberapa skenario untuk menguji kesesuaian respon daripada sistem.

5.2.1 Uji Coba Fungsionalitas

Uji coba fungsionalitas merupakan tahap pengujian yang dilakukan terhadap jalannya fungsi-fungsi utama pada sistem yang dibangun. Pengujian dilakukan terhadap seluruh fungsi, baik itu terdapat di sisi aplikasi *mobile* pada pengguna, program pada mikrokontroler Arduino, aplikasi web, dan pada sisi *server*. Uji coba yang dilakukan akan dijelaskan sebagai berikut:

1. Melakukan perintah pengendali jarak jauh

Melakukan perintah pengendali jarak jauh merupakan salah satu fungsi utama sistem. Fungsi ini dilakukan untuk memberikan perintah pada pendingin ruangan dari pengendali pendingin ruangan jarak jauh dari setiap data yang diterima dari *server*.

2. Menentukan jumlah pengguna

Data titik letak pengguna yang didapat dari *Indoor Positioning System* kemudian diolah untuk mendapatkan jumlah pengguna yang berada di dalam ruangan.

3. Menentukan kondisi pendingin ruangan

Menentukan kondisi pendingin ruangan dilakukan saat data jumlah pengguna telah ditentukan oleh *server*. Selanjutnya data ini dikirimkan pada mikrokontroler Arduino melalui jaringan WLAN dengan terkoneksi pada *access point*.

4. Menentukan suhu ideal secara adaptif

Setelah kondisi pendingin ruangan didapatkan kemudian *server* melakukan penentuan suhu ideal dengan data jumlah orang dan data suhu udara yang didapatkan. Fungsionalitas ini berfungsi untuk memberikan suhu ideal pada pendingin ruangan.

5.2.1.1 Uji Coba Melakukan Perintah Pengendali Jarak Jauh

Uji coba ini dilakukan dengan melakukan perintah pengendali jarak jauh dari mikrokontroler Arduino dengan memperoleh data

yang dikirimkan oleh *server* melalui jaringan Wi-Fi. Uji coba ini terbagi menjadi empat bagian, yaitu:

a. Melakukan perintah menghidupkan pendingin ruangan

Bagian pertama dari uji coba melakukan perintah pengendali jarak jauh adalah mengirimkan perintah untuk menghidupkan pendingin ruangan.

b. Melakukan perintah mematikan pendingin ruangan

Bagian kedua dari uji coba ini adalah melakukan uji coba perintah untuk mematikan pendingin ruangan.

c. Melakukan perintah menurunkan suhu pendingin ruangan

Bagian ketiga dari uji coba untuk melakukan perintah pengendali jarak jauh adalah melakukan perintah untuk menurunkan suhu pada pendingin ruangan.

d. Melakukan perintah menaikkan suhu pendingin ruangan

Tahap uji coba ini untuk memberi perintah pada pendingin ruangan untuk menaikkan suhu pada pendingin ruangan.

5.2.1.1.1 Melakukan Perintah Menghidupkan Pendingin Ruangan.

Uji coba ini dilakukan dengan mengirimkan data perintah dari *server* kepada mikrokontroler Arduino. Setelah data dikirimkan oleh *server* maka mikrokontroler akan mengeksekusi perintah yang dikirimkan. Data yang dikirimkan oleh *server* pada mikrokontroler Arduino berupa karakter 'H'. Gambar 5.5 merupakan tampilan dari *console* pada aplikasi *server* dalam melakukan pengiriman data perintah karakter untuk menghidupkan pendingin ruangan. Pengirim data perintah berupa karakter 'H' dikarenakan karakter tersebut sudah bisa menginisialisasikan suatu perintah tanpa harus mengirim sebuah kalimat perintah seperti "Menghidupkan pendingin ruangan". Hal tersebut juga mengurangi banyaknya jumlah data yang dikirimkan, oleh karena itu hanya diinisialisasikan dengan karakter tersebut.

```

Main (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jun 18, 2014 2:43:12 AM)
Perintah : N
[Informasi] Jumlah user : 0 Power AC : false Suhu : 25 Suhu Luar : 23
Successfully Connected to the database!
Updated successfully
Perintah : H
[Informasi] Jumlah user : 1 Power AC : true Suhu : 25 Suhu Luar : 23
Perintah : X
Successfully Connected to the database!
Updated successfully
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 23
Perintah : N
  
```

Gambar 5.5. Tampilan *server* saat mengirim perintah menghidupkan pendingin ruangan.

Setelah pengiriman data melalui jaringan *wireless* berhasil dilakukan, selanjutnya mikrokontroler akan melakukan eksekusi terhadap perintah data yang didapatkan. Gambar 5.6 merupakan tampilan hasil dari *serial port* pada Arduino IDE untuk memonitor data yang telah diterima oleh mikrokontroler.

```

COM29
connected
Temperature : 23 C
connected
Temperature : 23 C
Perintah : N
connected
Temperature : 23 C
Perintah : N
connected
Temperature : 23 C
Power ACPerintah : H
connected
Temperature : 24 C
Suhu naik 1 derajatPerintah : X
connected
Temperature : 23 C
  
```

Gambar 5.6. Tampilan dari *serial port* pada Arduino IDE.

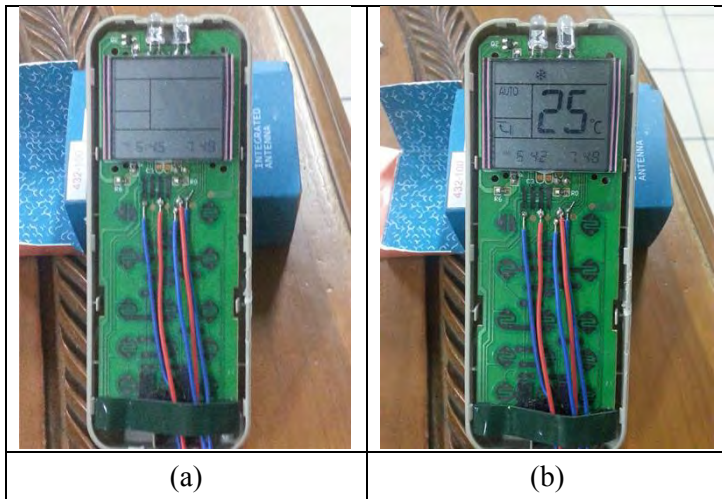
Pada gambar tersebut menunjukkan bahwa mikrokontroler Arduino sudah menerima data berupa karakter ‘H’ yang dikirimkan oleh *server*. Setelah itu mikrokontroler melakukan eksekusi perintah pada pengendali pendingin ruangan jarak jauh dengan memberi nilai digital pada dua *pin* digital yang masing-masing saling terhubung pada lingkaran tembaga pengendali pendingin ruangan jarak jauh.

Tabel 5.1. Prosedur Uji Coba Melakukan Perintah Menghidupkan Pendingin Ruangan pada Pengendali Jarak Jauh.

ID	UJ-01
Nama	Uji Coba Melakukan Perintah Menghidupkan Pendingin Ruangan pada Pengendali Jarak Jauh
Tujuan Uji Coba	Menguji fungsionalitas untuk menghidupkan pendingin ruangan.
Kondisi Awal	<i>Server</i> dan mikrokontroler Arduino terkoneksi pada jaringan yang sama, dan masing-masing saling menjalankan program.
Skenario	<i>Server</i> dijalankan untuk mengirimkan data pada mikrokontroler Arduino.
Masukan	-
Keluaran	Pendingin ruangan berhasil dijalankan.
Hasil uji coba	Berhasil

Sesuai yang dijelaskan pada Tabel 5.1, uji coba dilakukan dengan menjalankan program pada *server* yang kemudian program tersebut mengirimkan data untuk menghidupkan pendingin ruangan. *Output* yang diharapkan adalah pendingin ruangan dapat menyala.

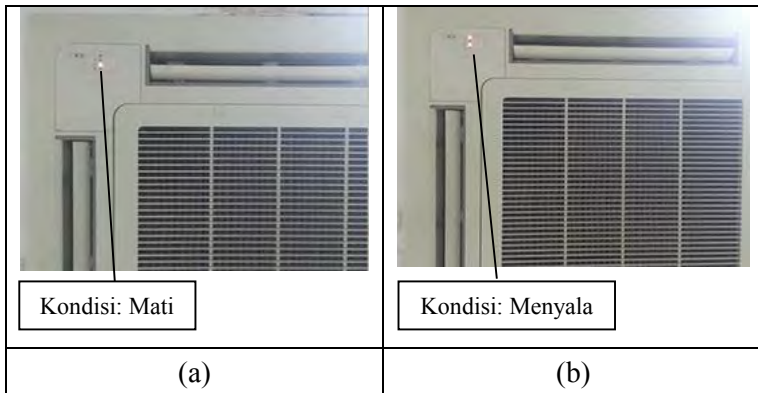
Apabila mikrokontroler Arduino sudah memberikan nilai digital pada *pin* digital yang terhubung dengan pengendali pendingin ruangan jarak jauh, maka pengendali ruangan yang sebelumnya tidak menyala akan menyala. Pada Gambar 5.7 menunjukkan tampilan dari pengendali pendingin ruangan jarak jauh. Di sebelah kiri pada gambar tersebut merupakan tampilan sebelum pemberian nilai digital, dan terdapat di sebelah kanan dari gambar tersebut adalah tampilan dari pengendali pendingin ruangan jarak jauh yang sudah mendapatkan data perintah untuk menghidupkan dan sudah melakukan eksekusi untuk menghidupkan pendingin ruangan.



Gambar 5.7. Tampilan pengendali pendingin jarak jauh sebelum (a) dan sesudah menerima perintah (b).

Pada Gambar 5.8 merupakan tampilan dari pendingin ruangan sebelum dan sesudah menerima perintah dari pengendali pendingin ruangan jarak jauh untuk menghidupkan. Di sebelah kiri pada gambar tersebut menunjukan tampilan dari pendingin sebelum dikirimkan perintah, terlihat hanya terdapat satu lampu

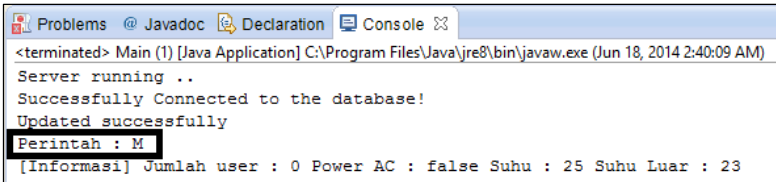
indikator yang menyala. Hal tersebut menandakan bahwa pendingin ruangan tidak menyala. Sebaliknya, di sebelah kanan dari gambar menunjukkan tampilan dari pendingin ruangan yang sudah menerima perintah untuk menghidupkan. Dua lampu indikator pada pendingin ruangan menyala, hal ini menunjukkan bahwa pendingin ruangan sudah menyala.



Gambar 5.8. Tampilan dari pendingin ruangan sebelum (a) dan sudah menerima perintah (b).

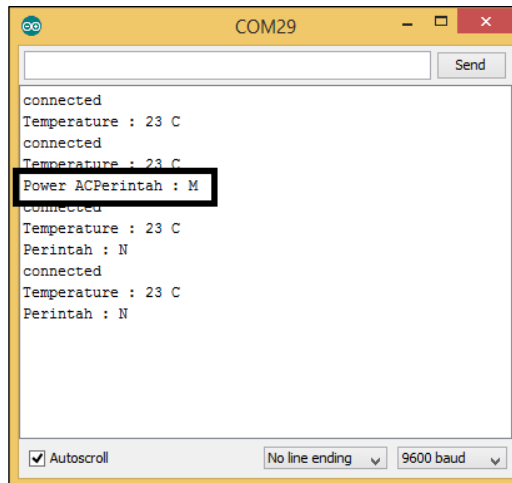
5.2.1.1.2 Melakukan Perintah Mematikan Pendingin Ruangan.

Uji coba ini dilakukan dengan mengirimkan data pada mikrokontroler Arduino. Selanjutnya, data yang dikirimkan oleh *server* tersebut akan dieksekusi secara langsung oleh mikrokontroler. Data yang dikirimkan berupa karakter 'M', dimana karakter ini menginisialisasikan perintah untuk mematikan pendingin ruangan.



Gambar 5.9. Tampilan *server* saat mengirim perintah mematikan pendingin ruangan.

Gambar 5.9 merupakan tampilan dari *console* pada aplikasi *server* dalam melakukan pengirim data perintah berupa karakter untuk mematikan pendingin ruangan. Tampak dalam gambar tersebut program pada *server* memberikan perintah berupa karakter 'M', kemudian perintah tersebut dikirimkan pada mikrokontroler Arduino agar dilakukan eksekusi mematikan pendingin ruangan menggunakan perintah yang telah dikirimkan.



Gambar 5.10. Tampilan dari memonitor perintah mematikan pendingin ruang menggunakan *serial port* pada Arduino IDE.

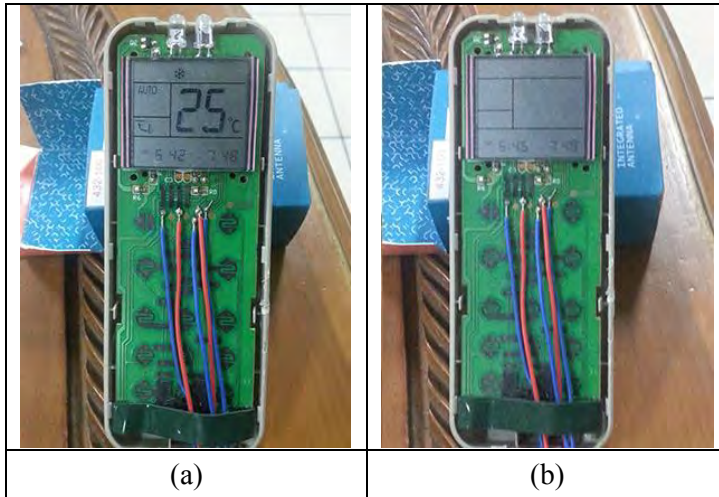
Data yang sudah diterima oleh mikrokontroler Arduino dapat dimonitor menggunakan *serial port* pada Arduino IDE, seperti pada Gambar 5.10. Pengecekan menggunakan *serial port* berfungsi untuk memastikan bahwa data yang dikirimkan oleh *server* pada mikrokontroler Arduino telah sampai dan data yang diterima berupa data karakter yang sama dengan data yang dikirimkan. Pada gambar tersebut menunjukkan bahwa mikrokontroler Arduino sudah menerima data karakter ‘M’ yang dikirimkan oleh *server* melalui jaringan WLAN.

Tabel 5.2. Prosedur Uji Coba Melakukan Perintah Mematikan Pendingin Ruangan pada Pengendali Jarak Jauh.

ID	UJ-02
Nama	Uji Coba Melakukan Perintah Mematikan Pendingin Ruangan pada Pengendali Jarak Jauh
Tujuan Uji Coba	Menguji fungsionalitas untuk mematikan pendingin ruangan.
Kondisi Awal	<i>Server</i> dan mikrokontroler Arduino terkoneksi pada jaringan yang sama, dan masing-masing saling menjalankan program.
Skenario	<i>Server</i> dijalankan untuk mengirimkan data perintah mematikan pendingin ruangan pada mikrokontroler Arduino.
Masukan	-
Keluaran	Pendingin ruangan berhasil dimatikan.
Hasil uji coba	Berhasil

Sesuai yang dijelaskan pada Tabel 5.2, uji coba dilakukan dengan menjalankan program pada *server* yang kemudian program tersebut mengirimkan data untuk mematikan pendingin ruangan.

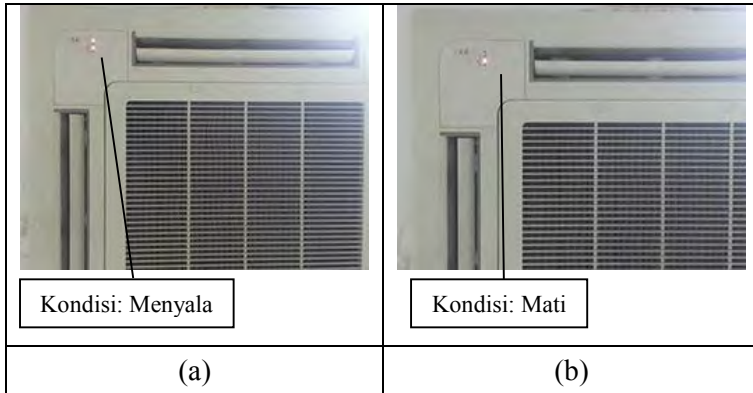
Output yang diharapkan adalah pendingin ruangan dapat dimatikan.



Gambar 5.11. Tampilan pengendali pendingin jarak jauh sebelum (a) dan sesudah menerima perintah mematikan pendingin ruangan (b).

Ketika mikrokontroler Arduino melakukan eksekusi perintah untuk mematikan pendingin ruangan, mikrokontroler Arduino memberikan nilai digital pada *pin* digital yang telah terhubung dengan pengendali pendingin ruangan jarak jauh, maka pengendali pendingin ruangan jarak jauh akan mati. Gambar 5.11 merupakan tampilan dari pengendali pendingin ruangan sebelum dan sesudah menerima data untuk mematikan pendingin ruangan. Sebelah kiri pada gambar tersebut menunjukkan tampilan pengendali pendingin ruangan sebelum dilakukan eksekusi untuk mematikan pendingin ruangan, layar pada pengendali pendingin ruangan jarak jauh terlihat masih menyala. Selanjutnya, bagian kanan dari gambar tersebut terlihat layar pada pengendali jarak jauh sudah tidak menyala. Hal tersebut menunjukkan bahwa

mikrokontroler Arduino sudah melakukan eksekusi perintah mematikan pendingin ruangan.



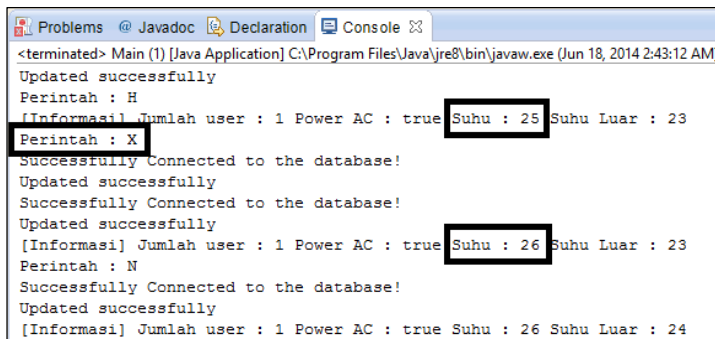
Gambar 5.12. Tampilan dari pendingin ruangan sebelum (a) dan sudah menerima perintah mematikan pendingin ruangan (b).

Pada Gambar 5.12 merupakan tampilan dari pendingin ruangan sebelum dan sesudah menerima perintah untuk mematikan pendingin ruangan dari pengendali pendingin ruangan jarak jauh. Terlihat disebelah kiri pada gambar menunjukkan tampilan pendingin ruangan dalam kondisi masih menyala sebelum menerima perintah, hal ini dapat dilihat dari dua lampu indikator yang masih menyala. Selanjutnya, pada bagian kanan gambar menunjukkan tampilan dari pendingin ruangan sesudah menerima perintah untuk mematikan. Hal ini bisa dibuktikan dengan terlihatnya lampu indikator yang menyala, pada bagian tersebut lampu indikator pada pendingin ruangan hanya satu yang menyala sedangkan lampu yang lainnya dalam posisi tidak menyala.

5.2.1.1.3 Melakukan Perintah Menaikkan Suhu Pendingin Ruangan.

Uji coba ini dilakukan dengan mengirimkan data perintah pada mikrokontroler Arduino. Data yang dikirimkan oleh program *server* berupa karakter 'X'. Gambar 5.13 merupakan tampilan *console* dari program *server* saat mengirimkan perintah menaikkan suhu pendingin ruangan, terlihat program pada *server* memberikan perintah 'X'. Karakter 'X' menginisialisasikan perintah untuk menurunkan suhu pada pendingin ruangan.

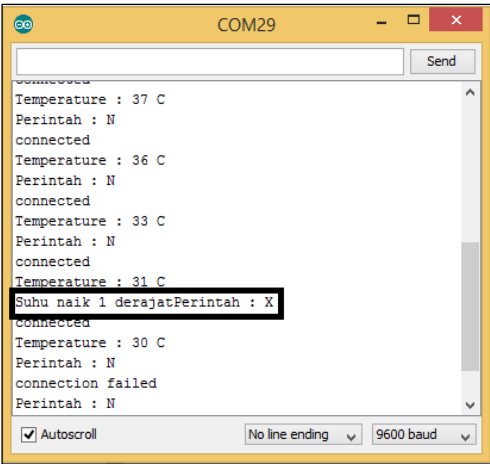
Setelah data didapatkan oleh mikrokontroler Arduino, selanjutnya mikrokontroler akan melakukan eksekusi terhadap perintah yang didapatkan. Pada Gambar 5.14 merupakan tampilan dari *serial port* pada Arduino IDE untuk melakukan monitor terhadap data yang sudah diterima. Hal ini yang menunjukkan bahwa mikrokontroler Arduino sudah menerima data berupa karakter 'X' yang dikirimkan. Selanjutnya data tersebut menginisialisasikan perintah melakukan perintah untuk menaikkan suhu ruangan dengan memberi nilai digital pada *pin* digital yang tersambung pada pengendali pendingin ruangan jarak jauh.



```

<terminated> Main (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jun 18, 2014 2:43:12 AM)
Updated successfully
Perintah : H
[Informasi] Jumlah user : 1 Power AC : true Suhu : 25 Suhu Luar : 23
Perintah : X
Successfully Connected to the database!
Updated successfully
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 23
Perintah : N
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 24
  
```

Gambar 5.13. Tampilan *server* saat mengirim perintah menaikkan suhu pendingin ruangan.



Gambar 5.14. Tampilan dari memonitor perintah menaikkan suhu pendingin ruang menggunakan serial port pada Arduino IDE.

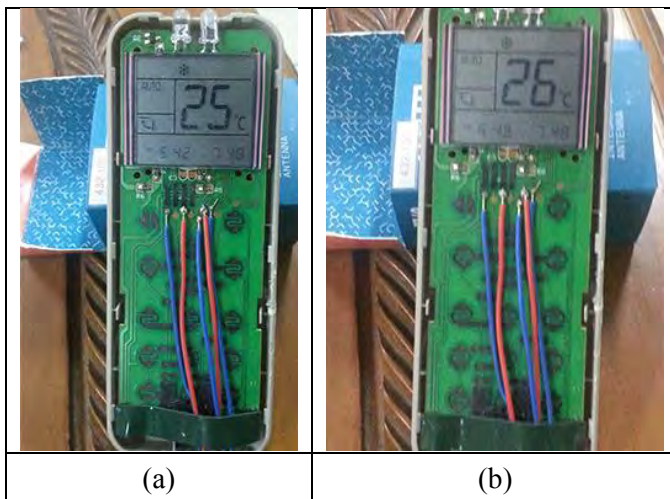
Sesuai yang dijelaskan pada Tabel 5.3, uji coba dilakukan dengan menjalankan program pada *server* yang kemudian program tersebut mengirimkan data untuk menurunkan suhu pendingin ruangan pada mikrokontroler Arduino. *Output* yang diharapkan adalah suhu pendingin ruangan dapat diturunkan.

Tabel 5.3. Prosedur Uji Coba Melakukan Perintah Menaikkan Suhu Pendingin Ruangan pada Pengendali Jarak Jauh.

ID	UJ-03
Nama	Uji Coba Melakukan Perintah Menaikkan Suhu Pendingin Ruangan pada Pengendali Jarak Jauh
Tujuan Uji Coba	Menguji fungsionalitas untuk menaikkan suhu pendingin ruangan.
Kondisi Awal	<i>Server</i> dan mikrokontroler Arduino terkoneksi pada jaringan yang sama, dan masing-masing saling menjalankan program.

Skenario	Server dijalankan untuk mengirimkan data perintah menaikkan suhu pendingin ruangan pada mikrokontroler Arduino.
Masukan	-
Keluaran	Suhu pendingin ruangan berhasil dinaikkan.
Hasil uji coba	Berhasil

Saat melakukan eksekusi perintah untuk menaikkan suhu pendingin ruangan, mikrokontroler Arduino memberi nilai digital pada *pin* digital yang terhubung dengan pengendali pendingin ruangan jarak jauh. Setelah pemberian nilai digital dilakukan, maka pengendali pendingin ruangan akan memberikan perintah terhadap pendingin ruangan untuk menurunkan suhu.



Gambar 5.15. Tampilan pengendali pendingin jarak jauh sebelum (a) dan sesudah menerima perintah menaikkan suhu pendingin ruangan (b).

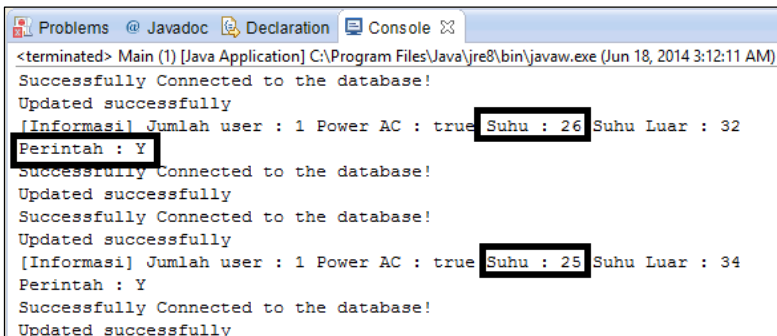
Pada Gambar 5.15 merupakan tampilan dari pengendali pendingin ruangan jarak jauh yang menunjukkan sebelum dan

sesudah pemberian nilai digital. Bagian di sebelah kiri gambar menunjukkan tampilan dari pengendali pendingin ruangan jarak jauh sebelum mikrokontroler Arduino menerima data perintah untuk menurunkan suhu pendingin ruangan. Di bagian kanan dari gambar tersebut menunjukkan bahwa mikrokontroler Arduino sudah melakukan eksekusi perintah, terlihat perbedaan angka sebelum dan sesudah yang menunjukkan suhu pada pendingin ruangan.

5.2.1.1.4 Melakukan Perintah Menurunkan Pendingin Ruangan.

Uji coba ini dilakukan dengan mengirimkan data perintah karakter kepada mikrokontroler Arduino. Data yang dikirimkan oleh *server* berupa karakter 'Y'. Pada Gambar 5.16 merupakan tampilan dari *console* dalam program *server*, saat mengirimkan perintah berupa karakter 'Y' kepada mikrokontroler Arduino.

Data perintah karakter yang dikirimkan oleh *server* menginisialisasikan perintah yang berbeda-beda. Namun, pada proses ini karakter 'Y' yang dikirimkan oleh *server* kepada mikrokontroler Arduino menginisialisasikan perintah untuk menurunkan suhu pada pendingin ruangan.

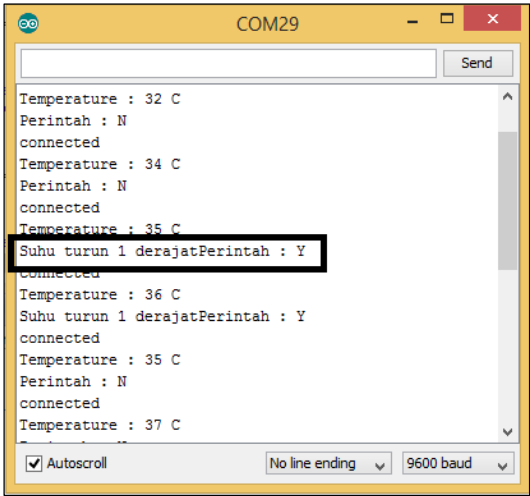


```

Problems  @ Javadoc  Declaration  Console  X
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jun 18, 2014 3:12:11 AM)
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 32
Perintah : Y
Successfully Connected to the database!
Updated successfully
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 25 Suhu Luar : 34
Perintah : Y
Successfully Connected to the database!
Updated successfully
  
```

Gambar 5.16. Tampilan *server* saat mengirim perintah menurunkan suhu pendingin ruangan.

Setelah pengiriman data berhasil dilakukan, selanjutnya mikrokontroler akan mengeksekusi perintah dengan menuliskan nilai digital pada *pin* digital yang tersambung pada pengendali pendingin ruangan jarak jauh.



Gambar 5.17. Tampilan dari memonitor perintah menurunkan suhu pendingin ruang menggunakan *serial port* pada Arduino IDE.

Pada Gambar 5.17 merupakan hasil dari memonitor data yang diterima oleh mikrokontroler Arduino melalui *serial port*. Dalam gambar tersebut menunjukkan bahwa mikrokontroler Arduino sudah menerima data berupa karakter ‘Y’ yang dikirimkan oleh *server*.

Tabel 5.4. Prosedur Uji Coba Melakukan Perintah Menurunkan Suhu Pendingin Ruangan pada Pengendali Jarak Jauh.

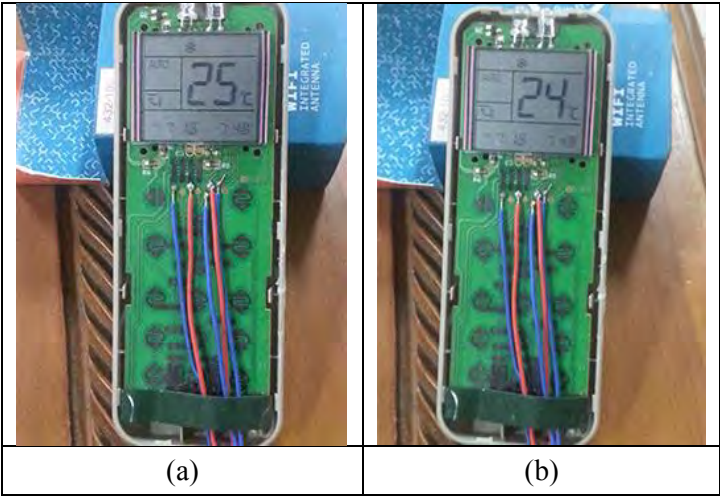
ID	UJ-04
Nama	Uji Coba Melakukan Perintah Menurunkan Suhu Pendingin Ruangan pada Pengendali Jarak Jauh

Tujuan Uji Coba	Menguji fungsionalitas untuk menurunkan suhu pendingin ruangan.
Kondisi Awal	<i>Server</i> dan mikrokontroler Arduino terkoneksi pada jaringan yang sama, dan masing-masing saling menjalankan program.
Skenario	<i>Server</i> dijalankan untuk mengirimkan data perintah menurunkan suhu pendingin ruangan pada mikrokontroler Arduino.
Masukan	-
Keluaran	Suhu pendingin ruangan berhasil diturunkan.
Hasil uji coba	Berhasil

Sesuai yang dijelaskan pada Tabel 5.4, uji coba dilakukan dengan menjalankan program pada *server* yang kemudian program tersebut mengirimkan data perintah untuk menurunkan suhu pendingin ruangan. *Output* yang diharapkan adalah suhu pada pendingin ruangan dapat diturunkan.

Selanjutnya, mikrokontroler Arduino melakukan eksekusi perintah dengan memberikan nilai digital pada *pin* digital yang tersambung dengan pengendali pendingin ruangan jarak jauh.

Pada Gambar 5.18 menunjukkan tampilan dari pengendali pendingin ruangan jarak jauh sebelum dan sesudah dilakukan eksekusi perintah. Di sebelah kanan pada gambar tersebut dapat dilihat perubahan angka suhu pada pendingin ruangan, hal ini menunjukan bahwa suhu pendingin ruangan berhasil diturunkan.



Gambar 5.18. Tampilan pengendali pendingin jarak jauh sebelum (a) dan sesudah menerima perintah menurunkan suhu pendingin ruangan (b).

5.2.1.2 Uji Coba Menentukan Jumlah Pengguna

Uji coba ini dilakukan dengan mendapatkan jumlah pengguna yang berada di dalam ruangan dengan data titik letak x dan y pengguna pada *database*. Data titik letak x dan y pengguna diperoleh dari *Indoor Positioning System* yang kemudian data tersebut disimpan dalam *database*, seperti pada Gambar 5.19.

ID_USER	X_USER	Y_USER	COLOR_USER
1	140.00	91.00	1

Gambar 5.19. Input data posisi pengguna.

Uji coba dilakukan dengan beberapa skenario, untuk lebih jelasnya dapat dilihat pada Tabel 5.5.

Tabel 5.5. Prosedur Uji Coba Menentukan Jumlah Pengguna.

ID	UJ-05
Nama	Uji Coba Menentukan Jumlah Pengguna
Tujuan Uji Coba	Menguji fitur untuk menentukan jumlah pengguna yang berada di dalam ruangan
Kondisi Awal	Telah memperoleh data titik letak posisi pengguna berupa koordinat x dan y dan ukuran ruangan laboratorium NCC.
Skenario 1	Pengguna <i>Dicky</i> berada di teras depan laboratorium NCC dan aplikasi dijalankan
Masukan	Data titik letak posisi pengguna.
Keluaran	Jumlah pengguna yang berada di dalam ruangan
Hasil uji coba	Berhasil
Skenario 2	Pengguna <i>Dicky</i> masuk ke dalam ruangan laboratorium NCC
Masukan	Data titik letak posisi pengguna.
Keluaran	Jumlah pengguna yang berada di dalam ruangan
Hasil uji coba	Berhasil
Skenario 3	Pengguna <i>Dicky</i> dan pengguna <i>Budi</i> masuk ke dalam ruangan laboratorium NCC
Masukan	Data titik letak posisi pengguna.
Keluaran	Jumlah pengguna yang berada di dalam ruangan
Hasil uji coba	Berhasil

Sesuai dengan prosedur pada Tabel 5.5, skenario pertama adalah pengguna Dicky berada di teras depan laboratorium NCC. Uji coba dilakukan ketika pengguna menggunakan IPS.



Gambar 5.20. Pengguna Dicky berada di teras depan laboratorium NCC.

Kemudian IPS dijalankan. Gambar 5.20 menunjukkan bahwa pengguna Dicky sedang berada di teras depan laboratorium NCC. Ponsel *pintar* akan mengirimkan data sinyal Wi-Fi kepada *server* untuk dilakukan pengolahan data yang menghasilkan data nilai titik x dan y posisi pengguna berada. *Server* melakukan *insert* data titik posisi pengguna pada *database*.

Pada Gambar 5.21 menunjukkan data titik letak pengguna di *database* yang sudah didapatkan dari IPS. Data ini berfungsi untuk dibandingkan dengan data ukuran ruangan.

ID_USER	X_USER	Y_USER	COLOR_USER
1	140.00	91.00	1

Gambar 5.21. Data posisi pengguna Dicky.

Server akan membandingkan data koordinat titik letak pengguna dengan ukuran dari ruangan laboratorium NCC. Selanjutnya, *server* akan menampilkan data jumlah pengguna yang

berada di dalam ruangan. Gambar 5.22 merupakan tampilan dari *console* pada *server* saat mendapatkan data jumlah pengguna.

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jun 19, 2014 12:35:57 PM)
Successfully Connected to the database!
Updated successfully
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 27
Perintah : N
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 26
Successfully Connected to the database!
Updated successfully
Perintah : M
[Informasi] Jumlah user : 0 Power AC : false Suhu : 26 Suhu Luar : 27
```

Gambar 5.22. Tampilan *server* saat mendapatkan data jumlah pengguna sebanyak nol pengguna.

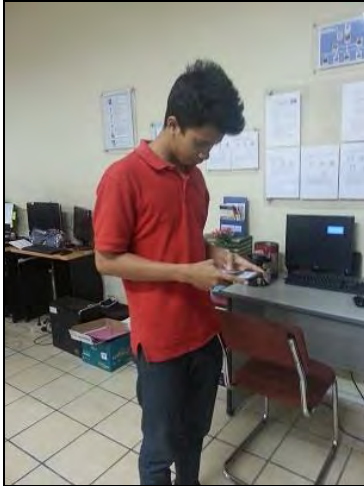


Gambar 5.23. Tampilan dari aplikasi web yang menampilkan informasi sistem.

Berdasarkan dengan data titik letak pengguna yang didapatkan di *database*, dapat dilihat pada Gambar 5.21. *Server*

dapat menentukan jumlah pengguna. Kemudian, data titik letak pengguna Dicky dan jumlah pengguna ditampilkan. Gambar 5.23 merupakan tampilan informasi dari sistem yang diakses menggunakan *browser*, menampilkan *map* yang berisikan letak pengguna Dicky dan jumlah pengguna yang berada di dalam ruangan.

Untuk skenario berikutnya, pengguna Dicky masuk ke dalam ruangan laboratorium NCC. Uji coba dimulai ketika pengguna menjalankan IPS. Selanjutnya, pengguna menjalankan IPS. Gambar 5.24 menunjukkan bahwa pengguna Dicky sedang berada di dalam ruangan laboratorium NCC dan menjalankan IPS. Data kekuatan sinyal Wi-Fi dikirim dari ponsel *pintar* pengguna kepada *server* untuk dilakukan pengolahan data yang lebih lanjut untuk mendapatkan data nilai letak posisi pengguna, yang kemudian disimpan di *database*. *Server* mendapatkan data titik letak pengguna berupa titik x dan y yang diperoleh dari tabel pada *database*, dapat dilihat pada Gambar 5.25.

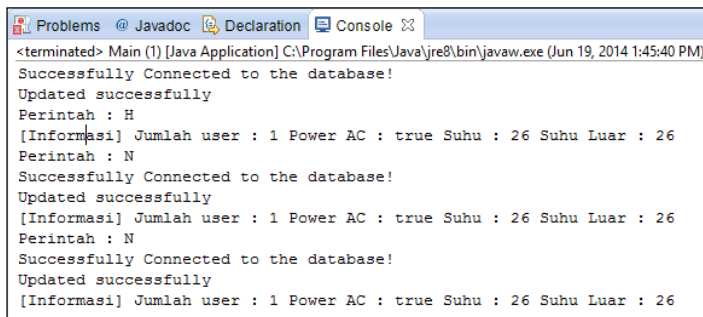


Gambar 5.24. Pengguna Dicky berada di dalam ruangan laboratorium NCC.

ID_USER	X_USER	Y_USER	COLOR_USER
1	140.00	284.00	1

Gambar 5.25. Data posisi pengguna Dicky di dalam ruangan.

Data koordinat letak posisi pengguna yang sudah didapatkan oleh *server* dibandingkan dengan data ukuran dari ruangan laboratorium. Gambar 5.26 merupakan tampilan dari *console* pada *server* mendapatkan data nilai jumlah pengguna. Pada gambar tersebut menunjukkan bahwa *server* memperoleh data jumlah pengguna sebanyak satu pengguna yang berada di dalam ruangan laboratorium NCC.



```

Problems @ Javadoc Declaration Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jun 19, 2014 1:45:40 PM)
Successfully Connected to the database!
Updated successfully
Perintah : H
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 26
Perintah : N
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 26
Perintah : N
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 26

```

Gambar 5.26. Tampilan *server* saat mendapatkan data jumlah pengguna sebanyak satu pengguna.

Berdasarkan data titik letak pengguna yang diperoleh dari *database*, data jumlah pengguna yang berada dalam ruangan dapat ditentukan. Kemudian data jumlah pengguna dan titik letak pengguna yang digambarkan pada *map* dapat dilihat menggunakan *browser*. Gambar 5.27 merupakan tampilan dari pengaksesan pada *browser* untuk melihat letak posisi pengguna di dalam denah dan informasi terkini dari sistem yang dibangun.



Gambar 5.27. Tampilan dari aplikasi web yang menampilkan informasi sistem (Bagian 2).

Skenario terakhir, pengguna Dicky dan pengguna Budi masuk ke dalam ruangan laboratorium NCC. Uji coba dimulai ketika kedua pengguna menjalankan IPS. Kemudian, kedua pengguna tersebut menjalankan IPS. Gambar 5.28 menunjukkan bahwa pengguna Dicky dan pengguna Budi berada di dalam ruangan laboratorium NCC serta sedang menjalankan IPS. Data kekuatan sinyal Wi-Fi dikirim dari masing-masing ponsel *pintar* milik pengguna kepada *server* untuk dilakukan pengolahan data. Selanjutnya, data tersebut disimpan pada *database*, dapat dilihat pada Gambar 5.29.

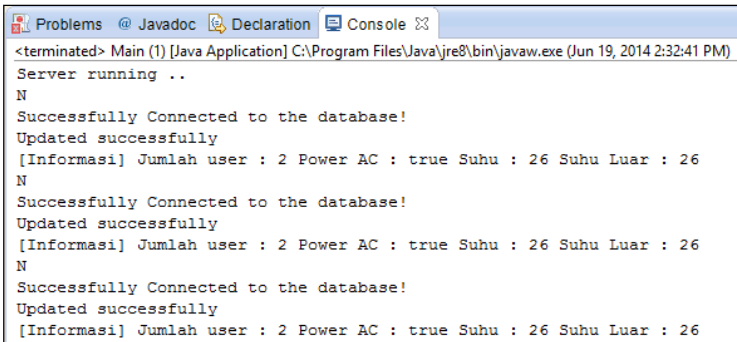


Gambar 5.28. Pengguna Dicky dan pengguna Budi berada di dalam ruangan laboratorium NCC.

ID_USER	X_USER	Y_USER	COLOR_USER
1	140.00	284.00	1
2	140.00	477.00	2

Gambar 5.29. Data posisi pengguna Dicky dan pengguna Budi di dalam ruangan.

Data yang telah didapatkan dari *database* oleh *server*, dibandingkan dengan data ukuran dari ruangan laboratorium NCC. Gambar 5.30 merupakan tampilan dari *console* pada program *server* yang mendapatkan data sebanyak dua pengguna berada di dalam ruangan laboratorium NCC.

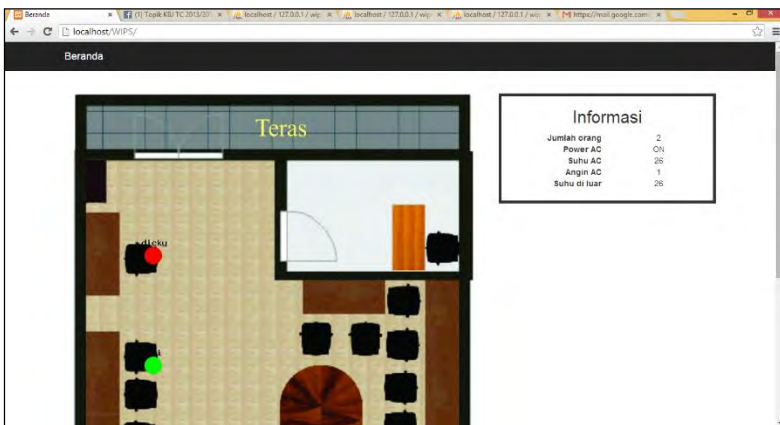


```

Problems @ Javadoc Declaration Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jun 19, 2014 2:32:41 PM)
Server running ..
N
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 2 Power AC : true Suhu : 26 Suhu Luar : 26
N
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 2 Power AC : true Suhu : 26 Suhu Luar : 26
N
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 2 Power AC : true Suhu : 26 Suhu Luar : 26

```

Gambar 5.30. Tampilan *server* saat mendapatkan data jumlah pengguna sebanyak dua pengguna.



Gambar 5.31. Tampilan dari aplikasi web yang menampilkan informasi sistem (Bagian 3).

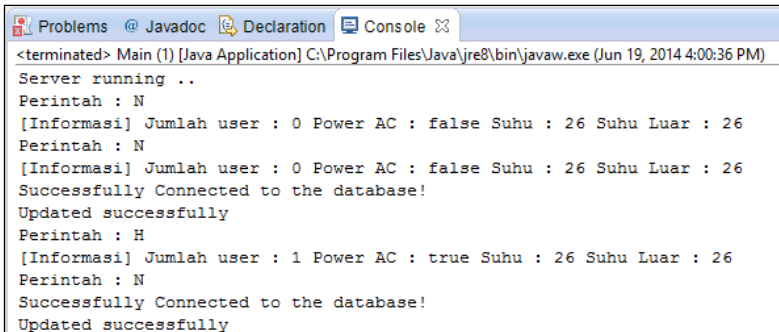
Berdasarkan titik letak masing-masing pengguna, dapat ditentukan jumlah pengguna yang berada di dalam ruangan. Data jumlah pengguna dan data informasi terkini dari sistem ditampilkan pada halaman web. Data tersebut dapat dilihat dengan mengakses menggunakan *browser*, hal ini ditunjukkan pada Gambar 5.31.

5.2.1.3 Uji Coba Menentukan Kondisi Pendingin Ruangan

Uji coba ini dilakukan dengan mendapatkan data jumlah pengguna yang berada di dalam ruangan. Setelah *server* memperoleh data jumlah pengguna, maka kondisi pendingin ruangan ditentukan. Berdasarkan prosedur pada Tabel 5.6, uji coba dilakukan dengan mendapatkan data jumlah pengguna yang berada di dalam ruangan. Kemudian, data tersebut dijadikan pertimbangan untuk menentukan kondisi pendingin ruangan. Kondisi pendingin ruangan dapat berubah-ubah terkait dengan perubahan jumlah data pengguna yang berada di dalam ruangan.

Tabel 5.6. Prosedur Uji Coba Menentukan Kondisi Pendingin Ruangan.

ID	UJ-06
Nama	Uji Coba Menentukan Kondisi Pendingin Ruangan
Tujuan Uji Coba	Menguji fitur untuk menentukan kondisi pendingin ruangan yang terdapat di dalam ruangan laboratorium NCC
Kondisi Awal	Telah memperoleh data jumlah pengguna yang berada di dalam ruangan
Skenario	<i>Server</i> dijalankan untuk menentukan kondisi pendingin ruangan
Masukan	Data jumlah pengguna yang berada di dalam ruangan
Keluaran	Data kondisi pendingin ruangan
Hasil uji coba	Berhasil



```

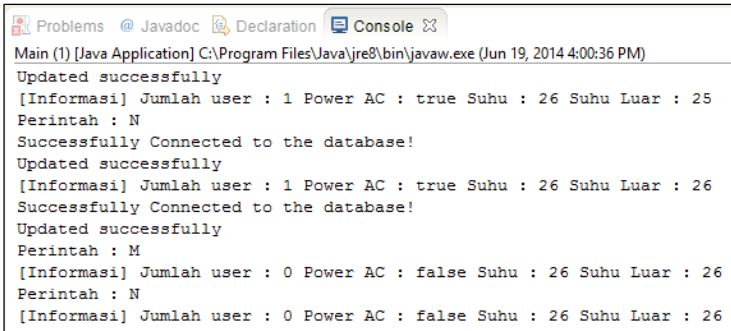
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jun 19, 2014 4:00:36 PM)
Server running ..
Perintah : N
[Informasi] Jumlah user : 0 Power AC : false Suhu : 26 Suhu Luar : 26
Perintah : N
[Informasi] Jumlah user : 0 Power AC : false Suhu : 26 Suhu Luar : 26
Successfully Connected to the database!
Updated successfully
Perintah : H
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 26
Perintah : N
Successfully Connected to the database!
Updated successfully

```

Gambar 5.32. Tampilan dari *server* saat menentukan kondisi pendingin ruangan dinyalakan.

Gambar 5.32 merupakan tampilan dari *console* pada program *server* saat menentukan kondisi pendingin ruangan dinyalakan. Terdapat pada gambar tersebut jumlah pengguna yang berada di dalam ruangan, data pengguna sebelumnya berjumlah nol dan pada selang waktu berikutnya terdapat pengguna yang masuk ke dalam ruangan. Hal ini yang membuat *server* menentukan kondisi pendingin ruangan untuk dinyalakan. Apabila kondisi pendingin ruangan sudah ditentukan, maka selanjutnya perintah dapat dikirimkan kepada mikrokontroler Arduino.

Pada Gambar 5.33 merupakan tampilan *console* dari program *server* saat menentukan kondisi pendingin ruangan dimatikan. Terlihat pada gambar tersebut jumlah pengguna yang berada di dalam ruangan dalam selang waktu tertentu, dari yang berjumlah sebanyak satu pengguna menjadi tidak ada pengguna yang berada di dalam ruangan. Hal ini yang membuat *server* menentukan kondisi pendingin ruangan untuk dimatikan.



```

Problems @ Javadoc Declaration Console
Main (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jun 19, 2014 4:00:36 PM)
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 25
Perintah : N
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 26
Successfully Connected to the database!
Updated successfully
Perintah : M
[Informasi] Jumlah user : 0 Power AC : false Suhu : 26 Suhu Luar : 26
Perintah : N
[Informasi] Jumlah user : 0 Power AC : false Suhu : 26 Suhu Luar : 26

```

Gambar 5.33. Tampilan dari *server* saat menentukan kondisi pendingin ruangan dimatikan.

5.2.1.4 Uji Coba Menentukan Suhu Ideal secara Adaptif

Uji coba ini dilakukan dengan mendapatkan data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan. Pada sistem ini seharusnya menggunakan sebanyak sepuluh pengguna yang dapat mempengaruhi penentuan suhu ideal secara adaptif. Namun, untuk uji coba digantikan dengan dua orang.

Setelah *server* memperoleh data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan, maka suhu ideal dapat ditentukan. Uji coba ini terbagi menjadi tiga bagian, yaitu:

- a. Menentukan suhu ideal secara adaptif dengan nilai suhu di luar ruangan yang memiliki rentang 28° hingga 33° Celcius

Bagian pertama dari uji coba ini adalah menentukan suhu ideal di dalam ruangan dengan data nilai di luar ruangan yang memiliki rentang antara 28° Celcius hingga 33° Celcius.

- b. Menentukan suhu ideal secara adaptif dengan nilai suhu di luar ruangan dibawah 28° Celcius

Bagian kedua dari uji coba ini melakukan penentuan suhu ideal di dalam ruangan dengan data nilai suhu di luar ruangan dibawah 28° Celcius.

- c. Menentukan suhu ideal secara adaptif dengan nilai suhu di luar ruangan di atas 33° Celcius

Tahap uji coba ini untuk menentukan suhu ideal di dalam ruangan dengan data nilai suhu di luar ruangan diatas 33° Celcius.

5.2.1.4.1 Uji Coba Menentukan Suhu Ideal secara Adaptif dengan Nilai Suhu yang Memiliki Rentang 28° hingga 33° Celcius

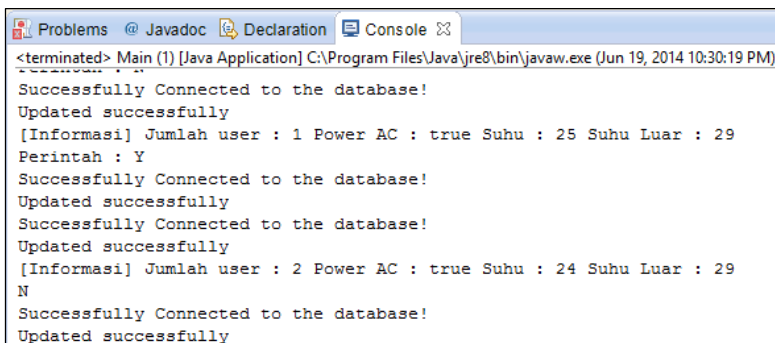
Uji coba ini dilakukan dengan mendapatkan data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan. Data nilai suhu yang digunakan dalam uji coba ini memiliki rentang antara 28° Celcius hingga 33° Celcius. Berdasarkan prosedur pada Tabel 5.1, uji coba dilakukan dengan mendapatkan data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan. Keduanya hal tersebut dapat mempengaruhi dalam menentukan suhu ideal secara adaptif. Data nilai suhu di luar ruangan diperoleh dari data yang dikirimkan oleh mikrokontroler Arduino kepada *server* melalui jaringan WLAN.

Tabel 5.7. Prosedur Uji Coba Menentukan Suhu Ideal secara Adaptif dengan Nilai Suhu yang Memiliki Rentang 28° hingga 33° Celcius.

ID	UJ-07
Nama	Uji Coba Menentukan Suhu Ideal secara Adaptif dengan Nilai Suhu yang Memiliki Rentang 28° hingga 33° Celcius
Tujuan Uji Coba	Menguji fitur untuk menentukan suhu ideal secara adaptif dengan nilai suhu dalam rentang 28° Celsius hingga 33° Celcius
Kondisi Awal	Telah memperoleh data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan
Skenario 1	Terdapat seorang pengguna yang berada di dalam ruangan labolatorium NCC dan

	selang waktu tertentu seorang pengguna lain masuk ke dalam ruangan
Masukan	Data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan
Keluaran	Suhu yang ideal di dalam ruangan
Hasil uji coba	Berhasil
Skenario 2	Terdapat dua pengguna yang berada di dalam ruangan labolatorium NCC dan selang waktu tertentu seorang pengguna meninggalkan ruangan
Masukan	Data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan
Keluaran	Suhu yang ideal di dalam ruangan
Hasil uji coba	Berhasil

Skenario yang pertama yaitu terdapat seorang pengguna yang berada di dalam ruangan laboratorium. Pada selang waktu tertentu seorang pengguna lainnya masuk ke dalam ruangan.



```

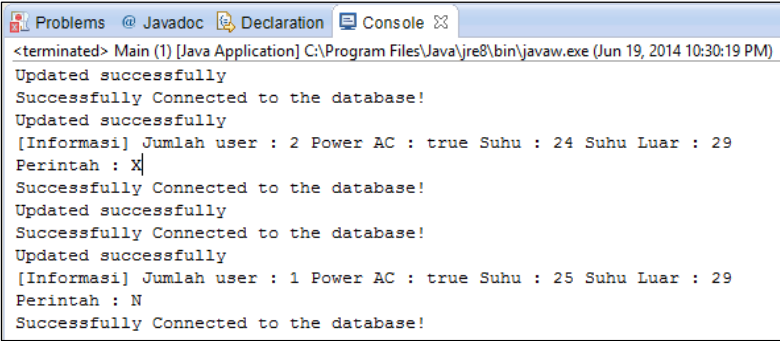
Problems @ Javadoc Declaration Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jun 19, 2014 10:30:19 PM)
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 25 Suhu Luar : 29
Perintah : Y
Successfully Connected to the database!
Updated successfully
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 2 Power AC : true Suhu : 24 Suhu Luar : 29
N
Successfully Connected to the database!
Updated successfully

```

Gambar 5.34. Tampilan dari *server* dalam menentukan suhu ideal dengan rentang nilai suhu antara 28° dan 33° C.

Pada Gambar 5.34 merupakan tampilan dari *console* pada program *server* yang menunjukkan perubahan nilai suhu pendingin ruangan. Data pengguna pada awalnya sebanyak satu pengguna, kemudian seorang pengguna masuk ke dalam ruangan sehingga suhu pada pendingin ruangan diturunkan.

Untuk skenario berikutnya, terdapat dua orang pengguna yang berada di dalam ruangan. Selanjutnya, dalam selang waktu tertentu seorang pengguna meninggalkan ruangan. Gambar 5.35 merupakan tampilan dari *console* pada program *server*. Dalam gambar tersebut terlihat data nilai suhu pendingin ruangan bertambah dari sebelumnya. Hal ini yang menunjukkan bahwa data nilai suhu pada pendingin ruangan bertambah seiring berkurangnya jumlah pengguna.



```

<terminated> Main (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jun 19, 2014 10:30:19 PM)
Updated successfully
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 2 Power AC : true Suhu : 24 Suhu Luar : 29
Perintah : X
Successfully Connected to the database!
Updated successfully
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 25 Suhu Luar : 29
Perintah : N
Successfully Connected to the database!
  
```

Gambar 5.35. Tampilan dari *server* dalam menentukan suhu ideal dengan rentang nilai suhu antara 28° dan 33° C (Bagian 2).

5.2.1.4.2 Uji Coba Menentukan Suhu Ideal secara Adaptif dengan Nilai Suhu dibawah 28° Celcius

Uji coba ini dilakukan dengan mendapatkan data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan. Data nilai suhu di luar ruangan yang digunakan pada uji coba ini adalah data nilai suhu dibawah 28°.

Tabel 5.8. Prosedur Uji Coba Menentukan Suhu Ideal secara Adaptif dengan Nilai Suhu Dibawah 28° Celcius.

ID	UJ-08
Nama	Uji Coba Menentukan Suhu Ideal secara Adaptif dengan Nilai Suhu dibawah 28° Celcius
Tujuan Uji Coba	Menguji fitur untuk menentukan suhu ideal secara adaptif dengan nilai suhu dibawah 28° Celcius
Kondisi Awal	Telah memperoleh data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan
Skenario 1	Terdapat seorang pengguna yang berada di dalam ruangan labolatorium NCC dan selang waktu tertentu seorang pengguna lain masuk ke dalam ruangan
Masukan	Data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan
Keluaran	Suhu yang ideal di dalam ruangan
Hasil uji coba	Berhasil
Skenario 2	Terdapat dua pengguna yang berada di dalam ruangan labolatorium NCC dan selang waktu tertentu seorang pengguna meninggalkan ruangan
Masukan	Data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan
Keluaran	Suhu yang ideal di dalam ruangan
Hasil uji coba	Berhasil

Sesuai dengan prosedur pada Tabel 5.8, skenario pertama adalah terdapat satu pengguna yang berada di dalam ruangan. Beberapa saat kemudian terdapat pengguna lain yang memasuki

ruangan. Gambar 5.36 merupakan tampilan *console* dari program *server* ketika menentukan suhu ideal secara adaptif dengan nilai suhu dibawah 28° Celcius. Pada gambar tersebut menunjukkan data nilai suhu pendingin ruangan ketika di ruangan tersebut terdapat seorang pengguna berkurang karena terdapat seorang pengguna yang memasuki ruangan labolatorium NCC. Hal ini menunjukkan bahwa suhu pada pendingin ruangan berubah secara adaptif terkait dengan jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan.

```

Problems @ Javadoc Declaration Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jun 19, 2014 11:50:44 PM)
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 27
Perintah : N
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 27
Perintah : Y
Successfully Connected to the database!
Updated successfully
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 2 Power AC : true Suhu : 25 Suhu Luar : 27
N
  
```

Gambar 5.36. Tampilan dari *server* dalam menentukan suhu ideal dengan nilai suhu dibawah 28° C (Bagian 1).

Untuk skenario berikutnya, terdapat dua orang pengguna yang berada di dalam ruangan. Selanjutnya, dalam selang waktu tertentu seorang pengguna meninggalkan ruangan. Pada Gambar 5.37 merupakan tampilan dari *console* pada program *server*. Gambar tersebut menunjukkan bahwa data nilai suhu pendingin ruangan menaik dari data nilai suhu pada pendingin ruangan sebelumnya, ketika terdapat seorang pengguna yang meninggalkan ruangan laboratorium NCC.

```

Problems @ Javadoc Declaration Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jun 19, 2014 11:50:44 PM)
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 2 Power AC : true Suhu : 25 Suhu Luar : 27
Perintah : X
Successfully Connected to the database!
Updated successfully
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 26 Suhu Luar : 27
Perintah : N
Successfully Connected to the database!

```

Gambar 5.37. Tampilan dari *server* dalam menentukan suhu ideal dengan nilai suhu dibawah 28° C (Bagian 2).

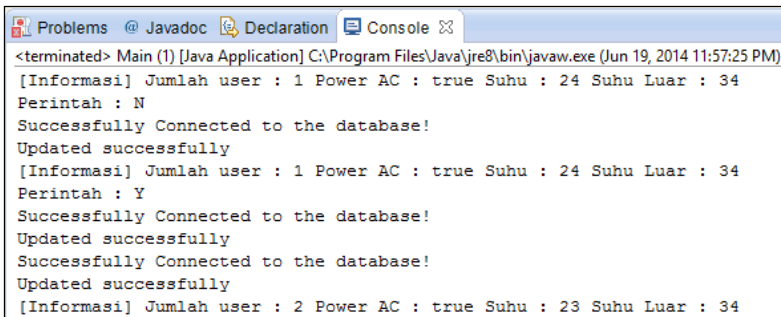
5.2.1.4.3 Uji Coba Menentukan Suhu Ideal secara Adaptif dengan Nilai Suhu diatas 33° Celcius

Uji coba ini dilakukan dengan mendapatkan data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan. Data nilai suhu yang digunakan pada uji coba ini adalah data dengan nilai suhu diatas 33° Celcius.

Tabel 5.9. Prosedur Uji Coba Menentukan Suhu Ideal secara Adaptif dengan Nilai Suhu Diatas 33° Celcius.

ID	UJ-09
Nama	Uji Coba Menentukan Suhu Ideal secara Adaptif dengan Nilai Suhu diatas 33° Celcius
Tujuan Uji Coba	Menguji fitur untuk menentukan suhu ideal secara adaptif dengan nilai suhu diatas 33° Celcius
Kondisi Awal	Telah memperoleh data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan

Skenario 1	Terdapat seorang pengguna yang berada di dalam ruangan labolatorium NCC dan selang waktu tertentu seorang pengguna lain masuk ke dalam ruangan
Masukan	Data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan
Keluaran	Suhu yang ideal di dalam ruangan
Hasil uji coba	Berhasil
Skenario 2	Terdapat dua pengguna yang berada di dalam ruangan labolatorium NCC dan selang waktu tertentu seorang pengguna meninggalkan ruangan
Masukan	Data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan
Keluaran	Suhu yang ideal di dalam ruangan
Hasil uji coba	Berhasil



```

Problems @ Javadoc Declaration Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jun 19, 2014 11:57:25 PM)
[Informasi] Jumlah user : 1 Power AC : true Suhu : 24 Suhu Luar : 34
Perintah : N
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 24 Suhu Luar : 34
Perintah : Y
Successfully Connected to the database!
Updated successfully
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 2 Power AC : true Suhu : 23 Suhu Luar : 34

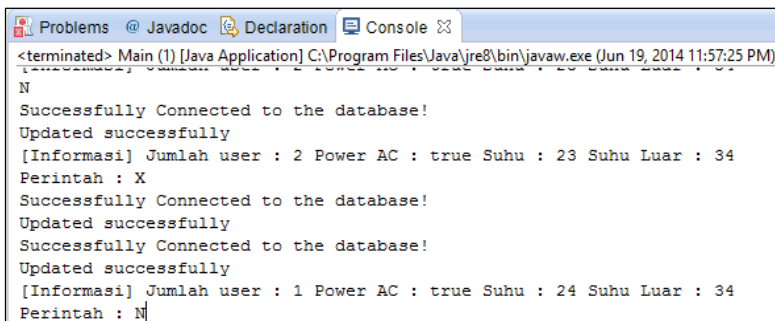
```

Gambar 5.38. Tampilan dari *server* dalam menentukan suhu ideal dengan nilai suhu diatas 33° C (Bagian 1).

Berdasarkan prosedur pada Tabel 5.9, uji coba dilakukan dengan mendapatkan data jumlah pengguna yang berada di dalam ruangan dan data nilai suhu di luar ruangan. Skenario pertama

adalah terdapat satu pengguna yang berada di dalam ruangan. Beberapa saat kemudian terdapat pengguna lain yang memasuki ruangan. Pada Gambar 5.38 menunjukkan bahwa data nilai suhu pendingin ruangan menurun sebanyak satu derajat, dikarenakan terdapat pengguna yang memasuki ruangan.

Skenario yang terakhir yaitu, terdapat dua pengguna yang berada di dalam ruangan serta didapatkan nilai suhu di luar ruangan sebesar 33° Celcius. Selanjutnya, dalam selang waktu tertentu salah satu pengguna meninggalkan ruangan. Gambar 5.39 menunjukkan bahwa data nilai suhu pendingin ruangan bertambah sebanyak satu derajat Celcius ketika terdapat seorang pengguna yang meninggalkan ruangan.



```

Problems @ Javadoc Declaration Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre8\bin\javaw.exe (Jun 19, 2014 11:57:25 PM)
N
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 2 Power AC : true Suhu : 23 Suhu Luar : 34
Perintah : X
Successfully Connected to the database!
Updated successfully
Successfully Connected to the database!
Updated successfully
[Informasi] Jumlah user : 1 Power AC : true Suhu : 24 Suhu Luar : 34
Perintah : N

```

Gambar 5.39. Tampilan dari *server* dalam menentukan suhu ideal dengan nilai suhu diatas 33° C (Bagian 2).

5.2.2 Uji Coba Performa

Pada bagian uji dilakukan uji coba performa terhadap sistem yang dibangun. Uji performa ini berfungsi untuk mengetahui perilaku dari sistem ketika dilakukan uji coba pada keadaan yang sebenarnya. Uji coba ini dilakukan untuk mengetahui kinerja dari sistem, *bandwidth consumption*, dan interval waktu pengiriman dan penerimaan data pada sistem yang dibangun.

5.2.2.1 Tingkat Kinerja pada Sistem

Uji coba performa ini menjelaskan kinerja dari sistem yang dibangun. Hasil yang diperoleh dari uji coba ini adalah sistem dapat memberikan perintah terhadap pengendali pendingin ruangan jarak jauh.

Pada Tabel 5.10 merupakan hasil dari uji coba menyalakan dan mematikan pendingin ruangan. Pada tabel tersebut menunjukkan bahwa sistem dapat memberikan perintah mematikan pendingin ruangan apabila tidak terdapat seseorang di dalam ruangan, dan dapat memberikan perintah untuk menyalakan pendingin ruangan apabila terdapat pengguna yang masuk ke dalam ruangan dari sebelumnya tidak terdapat pengguna di dalam ruangan.

Tabel 5.10. Hasil Uji Coba Menyalakan dan Mematikan Pendingin Ruangan.

Uji Coba Ke	Jumlah Pengguna (sebelum)	Jumlah Pengguna (sesudah)	Kondisi Pendingin Ruangan (sebelum)	Kondisi Pendingin Ruangan (sesudah)
1	0	1	Mati	Menyala
2	0	3	Mati	Menyala
3	1	0	Menyala	Mati
4	3	0	Menyala	Mati
5	6	0	Menyala	Mati

Tabel 5.11. Hasil Uji Coba Menentukan Suhu Ideal Secara Dinamis.

Jumlah Pengguna (sebelum)	Suhu Pendingin Ruangan (sebelum)	Suhu Lingkungan	Jumlah Pengguna (sesudah)	Perintah terhadap pendingin ruangan	Suhu Pendingin Ruangan (sesudah)
1	25° C	27° C	2	Menaikkan suhu	26° C
1	25° C	29° C	2	-	25° C

Jumlah Pengguna (sebelum)	Suhu Pendingin Ruangan (sebelum)	Suhu Lingkungan	Jumlah Pengguna (sesudah)	Perintah terhadap pendingin ruangan	Suhu Pendingin Ruangan (sesudah)
1	25° C	34° C	2	Menurunkan suhu	24° C
6	25° C	27° C	7	Menaikan Suhu	26° C
6	25° C	29° C	7	-	25° C
6	25° C	34° C	7	Menurunkan Suhu	24° C
9	25° C	27° C	10	Menaikan Suhu	26° C
9	25° C	29° C	10	-	25° C
9	25° C	34° C	10	Menurunkan Suhu	24° C
12	24° C	27° C	13	Menaikan Suhu	25° C
12	24° C	29° C	13	-	24° C
12	24° C	34° C	13	Menurunkan Suhu	23° C
15	24° C	27° C	16	Menaikan Suhu	25° C
15	24° C	29° C	16	-	24° C
15	24° C	34° C	16	Menurunkan Suhu	23° C

Dalam sistem ini ditetapkan batas atas dan batas bawah suhu pada pendingin ruangan. Batas atas yang ditetapkan untuk jumlah pengguna kurang dari sepuluh adalah 26° C, sedangkan batas bawah suhu pada pendingin ruangan untuk jumlah pengguna kurang dari sepuluh adalah sebesar 24° C. Selanjutnya, penetapan batas atas suhu pendingin ruangan untuk jumlah pengguna lebih besar dari sepuluh sebesar 25° C, dan untuk batas bawah yang ditetapkan jika jumlah pengguna lebih besar dari sepuluh adalah 23° C. Penetapan batas atas dan batas bawah berfungsi untuk menjaga suhu supaya tidak terlalu dingin ataupun terlalu panas. Suhu pada pendingin ruangan tidak dapat melebihi atau kurang dari batas yang telah ditentukan.

Suhu pada pendingin ruangan akan diturunkan jika jumlah pengguna lebih besar dari sepuluh. Hal ini membuat perbedaan

suhu pada pendingin ruangan sebesar satu derajat antara jumlah pengguna lebih besar dari sepuluh dan kurang dari sepuluh.

Kemudian, setelah dilakukan pemantauan terhadap suhu ideal di dalam ruangan, maka dilakukan penetapan batas atas dan batas bawah untuk suhu lingkungan, untuk batas atas yang ditetapkan sebesar 33°C dan batas bawah yang ditetapkan adalah sebesar 28° C. Apabila suhu lingkungan kurang dari 28° C maka suhu pada pendingin ruangan akan dinaikkan satu derajat, jika lebih besar dari 33° C maka suhu pada pendingin ruangan dinaikkan sebesar satu derajat, dan jika suhu berada diantara batas atas dan batas bawah maka suhu pendingin ruangan akan tetap.

Tabel 5.11 menunjukkan bahwa sistem telah dapat menentukan suhu ideal secara dinamis terkait dengan jumlah pengguna dan suhu lingkungan.

5.2.2.2 Interval Waktu Pengiriman dan Penerimaan Data

Uji coba performa ini menjelaskan mengenai waktu interval yang terjadi pada saat pengiriman data perintah dari *server* ke mikrokontroler Arduino dan pada saat mikrokontroler Arduino mengirim data sensor berisikan nilai suhu kepada *server*. Waktu data yang dihitung menggunakan satuan *millisecond* (ms).

Sistem ini memiliki waktu *delay* yang telah ditentukan, hal ini berfungsi untuk mencegah pengiriman perintah terhadap pendingin ruangan yang berulang-kali dengan kurun waktu yang singkat. Oleh karena itu diberikan waktu *delay*. Pada uji coba ini dilakukan dua kali dengan menggunakan dua waktu *delay* yang berbeda, yaitu: waktu *delay* yang pertama adalah sepuluh detik atau 10000 (ms) dan yang kedua adalah 30 detik atau 30000 (ms). Penggunaan waktu *delay* lebih dari 30 detik tidak dapat dilakukan karena dapat terjadi *connection reset* pada *server*.

Pada Tabel 5.12 merupakan data nilai interval waktu yang digunakan dalam pengiriman dan penerimaan data dengan menggunakan waktu *delay* sebesar sepuluh detik. Selanjutnya,

Tabel 5.13 menunjukkan data nilai interval waktu dengan menggunakan waktu *delay* sebesar 30 detik.

Tabel 5.12. Hasil uji coba interval waktu pengiriman dan penerimaan data menggunakan waktu *delay* sebesar sepuluh detik.

Data ke	Waktu Pengiriman	Waktu Penerimaan	Selisih Waktu (s)
1	20:53:32	20:53:45.803	13.803
2	20:53:48	20:54:01.732	13.732
3	20:54:03	20:54:16.604	13.604
4	20:54:19	20:54:32.954	13.954
5	20:54:34	20:54:47.781	13.781
6	20:54:50	20:55:03.652	13.652
7	20:55:06	20:55:19.472	13.472
8	20:55:21	20:55:34.842	13.842
9	20:55:37	20:55:50.649	13.649
10	20:55:52	20:56:05.499	13.499
Rata-Rata dengan Waktu <i>Delay</i> 10 (s)			3.699

Tabel 5.13. Hasil uji coba interval waktu pengiriman dan penerimaan data menggunakan waktu *delay* sebesar 30 detik.

Data ke	Waktu Pengiriman (ms)	Waktu Penerimaan (ms)	Selisih Waktu (ms)
1	21:03:59	21:04:32.376	33.376
2	21:04:34	21:05:07.699	33.699
3	21:05:10	21:05:43.774	33.774
4	21:05:46	21:06:19.831	33.831
5	21:06:21	21:06:54.920	33.920
6	21:06:57	21:07:30.498	33.498
7	21:07:33	21:08:06.609	33.609
8	21:08:08	21:08:41.698	33.698
9	21:08:44	21:09:17.752	33.752

Data ke	Waktu Pengiriman (ms)	Waktu Penerimaan (ms)	Selisih Waktu (ms)
10	21:09:19	21:09:52.828	33.828
Rata-Rata dengan Waktu <i>Delay</i> 30 (s)			3.699

5.2.2.3 *Bandwidth Consumption*

Uji performa ini menjelaskan penggunaan *bandwidth* dalam sistem ini untuk pengiriman dan penerimaan data antara *server* dengan mikrokontroler Arduino serta antara *server* dengan ponsel *pintar* pengguna. Tabel 5.14 merupakan hasil dari penggunaan *bandwidth* dalam sistem ini terkait dengan jumlah pengguna yang berada di dalam ruangan.

Tabel 5.14 . Hasil uji coba *bandwidth consumption*.

Luas Ruangan (m)	Estimasi Pengguna	<i>Bandwidth Consumption</i> (Mbit/s)
67.62	1	0.026
67.62	2	0.09
67.62	3	0.12
67.62	4	0.157
67.62	5	0.196
67.62	6	0.236
67.62	7	0.275
67.62	8	0.314
67.62	9	0.354
67.62	10	0.393
Rata-Rata (Mbit/s)		0.0393

BAB VI PENUTUP

Pada bab ini akan dibahas mengenai kesimpulan yang dapat diambil dari perancangan sistem, implementasi, hingga dengan hasil pengujian selama pengerjaan Tugas Akhir. Pada bab ini juga dapat menjawab pertanyaan yang dijabarkan pada Bab 1.

Pembuatan Tugas Akhir ini pasti memiliki beberapa kelebihan dan kekurangan dari hasil yang telah dicapai dari pembuatan sistem. Semua kelebihan dan kekurangan Tugas Akhir ini juga akan dijabarkan pada bab ini. Untuk memperbaiki semua kelebihan dan kekurangan dari sistem, akan dijelaskan pada subbab saran.

6.1. Kesimpulan

Berdasarkan dari hasil pengujian dari uji coba yang dilakukan selama pengerjaan Tugas Akhir ini, dapat diperoleh beberapa kesimpulan sebagai berikut:

1. Sistem telah mampu memerintah pengendali pendingin ruangan jarak jauh dan berintegrasi dengan *Indoor Positioning System*.
2. Sistem telah mampu memberikan suhu secara dinamis dengan integrasi jumlah orang di dalam ruangan dan suhu lingkungan.
3. Penggunaan *bandwidth* pada pengiriman dan penerimaan data antara mikrokontroler Arduino dengan *server* serta *server* dengan ponsel *pintar* pengguna memiliki kenaikan sejumlah 0.0393 (Mbit/s) seiring pertambahan jumlah pengguna sebanyak satu.
4. Interval waktu pengiriman dan penerimaan data antara *server* dan mikrokontroler Arduino dengan penggunaan waktu *delay* sebesar sepuluh detik dan 30 detik memiliki rata-rata yang sama. Dari hasil uji coba yang telah dilakukan, diperoleh rata-rata interval waktu pengiriman dan penerimaan data sebesar 3.699 detik.

6.2. Saran

Selama proses pengerjaan Tugas Akhir yang dimulai dari perancangan, implementasi, hingga uji coba, ditemukan beberapa kekurangan pada sistem. Adapun saran dari penulis yang dapat dilakukan untuk pengembangan lebih lanjut adalah sebagai berikut:

1. Meningkatkan keakuratan estimasi jumlah pengguna dengan menggunakan sensor pendeteksi gerakan.
2. Menggunakan protokol 802.15.4 dengan data *rate* yang rendah, seperti ZigBee sehingga dapat menghemat konsumsi tenaga yang digunakan.
3. Menggunakan sensor yang performanya lebih baik.

LAMPIRAN

Bagian ini berisikan lampiran dari buku Tugas Akhir ini sebagai pelengkap. Pada bagian ini terdapat beberapa potongan *source code* yang digunakan untuk membangun sistem ini. fungsi-fungsi yang digunakan untuk membangun aplikasi.

1. Fungsi hitungSuhu

Fungsi ini digunakan untuk mendapatkan data nilai suhu dari sensor suhu yang tersambung dengan mikrokontroler menggunakan IDE Arduino terdapat pada Kode Sumber A.1.

```
void hitungSuhu(){
    tempc = 0;
    for(int i = 0;i<=5;i++){
        temp = analogRead(tempPin);
        tempc += (5.0 * temp * 100.0)/1024.0;
        delay(90);
    }
    tempc = tempc/6;
    //debugging
    Serial.print("Temperature : ");
    Serial.print(tempc);
    Serial.print(" C");
    Serial.println();
    //send temperature to server
    client.println(tempc);
}
```

Kode Sumber A.1. Fungsi hitungSuhu.

2. Fungsi suhuNaik

Fungsi ini digunakan mikrokontroler Arduino untuk memberikan perintah pada pengendali pendingin ruangan jarak jauh untuk menaikkan suhu pendingin ruangan terdapat pada Kode Sumber A.2.

```
void suhuNaik()
{
    digitalWrite(3, HIGH);
    digitalWrite(5, HIGH);
    delay(1000);
    digitalWrite(3, LOW);
    digitalWrite(5, LOW);
    Serial.print("Suhu naik 1 derajat");
    Serial.println();
}
```

Kode Sumber A.2. Fungsi suhuNaik.

3. Fungsi suhuTurun

Fungsi ini digunakan mikrokontroler Arduino untuk memberikan perintah pada pengendali pendingin ruangan jarak jauh untuk menurunkan suhu pendingin ruangan terdapat pada Kode Sumber A.3.

```
void suhuTurun()
{
    digitalWrite(3, HIGH);
    digitalWrite(6, HIGH);
    delay(1000);
    digitalWrite(3, LOW);
}
```

```
digitalWrite(6, LOW);
Serial.print("Suhu turun 1 derajat");
Serial.println();
}
```

Kode Sumber A.3. Fungsi suhuTurun.

4. Fungsi *Power*

Fungsi ini digunakan mikrokontroler Arduino untuk memberikan perintah pada pengendali pendingin ruangan jarak jauh untuk menyalakan dan mematikan pendingin ruangan terdapat pada Kode Sumber A.4.

```
void Power()
{
    digitalWrite(2, HIGH);
    digitalWrite(6, HIGH);
    delay(1000);
    digitalWrite(2, LOW);
    digitalWrite(6, LOW);
    Serial.print("Power AC");
    Serial.println();
}
```

Kode Sumber A.4. Fungsi *Power*.

5. Class *User*

Class ini berfungsi untuk mengambil data pada *database* dan menentukan apakah pengguna berada di dalam ruangan terdapat pada Kode Sumber A.5.

```
package com.resource;
```

```

public class user {
    int id;
    double x,y;

    public user() {
    }

    public user(int id, double x, double y) {
        super();
        this.id = id;
        this.x = x;
        this.y = y;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public double getX() {
        return x;
    }

    public void setX(double x) {
        this.x = x;
    }

    public double getY() {
        return y;
    }

    public void setY(double y) {
        this.y = y;
    }

    public int countUser(int id,double x,double y)
    {
        int jumlahUser = 0;

        if(y > 91.0)

```

```

        {
            jumlahUser++;
        }

        return jumlahUser;
    }
}

```

Kode Sumber A.5. Class User.

6. Fungsi Pengaturan Suhu Adaptif

Fungsi ini digunakan *server* untuk pengaturan suhu adaptif yang berfungsi untuk menentukan perintah mematikan, menyalakan pendingin ruangan, menurunkan dan menaikkan suhu pendingin ruangan terdapat pada Kode Sumber A.6.

```

        if(user>0)
        {
            if(powerAC==false)
            {
                powerAC=true;

                db.updatePower(powerAC);
                clientOutput.writeBytes("H");

                System.out.println("Perintah : H");
            }
            else
            {
                if(user>1)
                {
                    if(suhuAC>24
&& suhuLuar<=33 && suhuLuar>=28)
                    {

                        suhuAC--;

                        System.out.println("Perintah : Y");

                        clientOutput.writeBytes("Y");

                        db.updateSuhu(suhuAC);

```

```

    }
    else
if(suhuAC<24 && suhuLuar<=33 && suhuLuar>=28)
    {
        suhuAC++;

        System.out.println("Perintah : X");
        clientOutput.writeBytes("X");
        db.updateSuhu(suhuAC);
    }
    else
if(suhuLuar>33 && (suhuAC==24 || suhuAC==25))
    {

        suhuAC--;

        System.out.println("Perintah : Y");

        clientOutput.writeBytes("Y");

        db.updateSuhu(suhuAC);

    }
    else
if(suhuLuar<28 && (suhuAC==23 || suhuAC==24))
    {
        suhuAC++;

        System.out.println("Perintah : X");

        clientOutput.writeBytes("X");

        db.updateSuhu(suhuAC);
    }
    else
if(suhuLuar<28 && (suhuAC==26))
    {

        suhuAC--;

        System.out.println("Perintah : Y");

        clientOutput.writeBytes("Y");

        db.updateSuhu(suhuAC);
    }

```

```

    }
    else
    {

        System.out.println("N");

        clientOutput.writeBytes("N");

    }

    db.updateSuhuLuar(suhuLuar);
    }
    else if(user<2)
    {
        if(suhuAC<=24
        && suhuLuar<=33 && suhuLuar>=28)
        {

            suhuAC++;

            System.out.println("Perintah : X");

            clientOutput.writeBytes("X");

            db.updateSuhu(suhuAC);
        }
        else
        if(suhuAC>25 && suhuLuar<=33 && suhuLuar>=28)
        {

            suhuAC--;

            System.out.println("Perintah : Y");

            clientOutput.writeBytes("Y");

            db.updateSuhu(suhuAC);
        }
        else
        if(suhuLuar>33 && (suhuAC==25 || suhuAC==26))
        {

            suhuAC--;

            System.out.println("Perintah : Y");

```



```
db.updatePower(powerAC);

System.out.println("Perintah : M");

clientOutput.writeBytes("M");
    }
    else
    {

System.out.println("Perintah : N");
        clientOutput.writeBytes("N");
    }
}
```

Kode Sumber A.6. Fungsi Pengaturan Suhu Adaptif.

BIODATA PENULIS



R Dicky Budi Aldyanto, dilahirkan di kota Madiun, Jawa Timur pada 13 April 1992. Penulis adalah anak kedua dari tiga bersaudara dan dibesarkan di kota Bekasi, Jawa Barat. Penulis menempuh pendidikan formal di SDN Pondok Kelapa 05 Pagi Jakarta (1998-2004), SMPN 252 Jakarta (2004-2007), SMAN 44 Jakarta (2007-2010). Pada tahun 2010, penulis memulai pendidikan di strata satu Jurusan Teknik

Informatika Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember, Surabaya, Jawa Timur dengan nomor induk mahasiswa 5110100036. Di jurusan Teknik Informatika, penulis mengambil bidang minat NCC (*Net Centric Computing*). Penulis juga aktif dalam organisasi kemahasiswaan seperti HMTC (Himpunan Mahasiswa Teknik Informatika). Penulis juga sempat beberapa kali menjadi asisten dosen, diantaranya Asisten Basis Data Lanjut, dan Topik Khusus Komputasi Berbasis Jaringan. Penulis dapat dihubungi melalui alamat email aldyanto.dicky@gmail.com atau dicky.aldyanto@yahoo.com.